scientific reports

Check for updates

OPEN

Enabling scalable single-cell transcriptomic analysis through distributed computing with Apache spark

Asif Adil[1,2,7]✉, Namrata Bhattacharya[3,4], Aadam[5], Naveed Jeelani Khan[6]✉ & Mohammed Asger[6]✉

As the field of single-cell genomics continues to develop, the generation of large-scale scRNA-seq datasets has become more prevalent. Although these datasets offer tremendous potential for shedding light on the complex biology of individual cells, the sheer volume of data presents significant challenges for management and analysis. Off late, to address these challenges, a new discipline, known as "big single-cell data science," has emerged. Within this field, a variety of computational tools have been developed to facilitate the processing and interpretation of scRNA-seq data. However, several of these tools primarily focus on the analytical aspect and tend to overlook the burgeoning data deluge generated by scRNA-seq experiments. In this study, we try to address this challenge and present a novel parallel analytical framework, scSPARKL, that leverages the power of Apache Spark to enable the efficient analysis of single-cell transcriptomic data. scSPARKL is fortified by a rich set of staged algorithms developed to optimize the Apache Spark's work environment. The tool incorporates six key operations for dealing with single-cell Big Data, including data reshaping, data preprocessing, cell/gene filtering, data normalization, dimensionality reduction, and clustering. By utilizing Spark's unlimited scalability, fault tolerance, and parallelism, the tool enables researchers to rapidly and accurately analyze scRNA-seq datasets of any size. We demonstrate the utility of our framework and algorithms through a series of experiments on real-world scRNA-seq data. Overall, our results suggest that scSPARKL represents a powerful and flexible tool for the analysis of single-cell transcriptomic data, with broad applications across the fields of biology and medicine.

Keywords Apache spark, ScRNA-seq, Normalization, Quality control, Big data

Single-cell RNA sequencing (scRNA-seq) has transformed our understanding of cellular biology by enabling the study of gene expression at the resolution of individual cells. Unlike bulk RNA-seq, which assumes cellular homogeneity within a tissue, scRNA-seq has revealed that cells within the same tissue can exhibit substantial heterogeneity. This cellular diversity plays a crucial role in driving phenotypic variation across tissues and biological systems[1]. An advantage of scRNA-seq over traditional bulk sequencing, is the identification of rare cells in the same tissues based on various cell features which were unknown earlier[2]. It also permits a more in-depth examination of processes such as development and differentiation of cells[3]. This enhancement in resolution, however, comes at the expense of increased technical noise and biases[4]. This implies that pre-processing, quality control, and normalization are essential for a thorough analysis of scRNA-seq data[5].

Additionally, the advancement in cell profiling techniques and rapid cost drops, has led to the sequencing millions of cells parallelly. Modern scRNA-seq technologies, such as those from 10x Genomics, routinely

[1]Department of Computer Sciences, Baba Ghulam Shah Badshah University, Rajouri, India. [2]Department of Pathology and Laboratory Medicine, School of Medicine, Indiana University Indianapolis, Indianapolis, IN, USA. [3]Department of Computer Science and Engineering, Indraprastha Institute of Information Technology, New Delhi, India. [4]Australian Prostate Cancer Research Center, Queensland University of Technology, Brisbane, Australia. [5]Department of Computer Science, Luddy School of Informatics, Indiana University Indianapolis, Indianapolis, IN, USA. [6]Department of Computer Science and Engineering, Model Institute of Engineering and Technology, Jammu, Jammu and Kashmir, India. [7]Department of Pathology and Laboratory Medicine, Indiana University Indianapolis, Indianapolis, IN, USA. ✉email: asifadil@bgsbu.ac.in; asif@iu.edu; naveed.cse@mietjammu.in; naveed1313@gmail.com; asger.cse@mietjammu.in

generate datasets containing hundreds of thousands of cells, each with expression profiles spanning over 20,000 genes[6]. The Illumina HiSeq, another next-generation sequencing platform, can alone generate 100 GBs of raw transcriptomics data[7]. This results in extremely sparse, high-dimensional matrices that demand considerable computational power for downstream analysis[8,9]. Preprocessing steps such as quality control, normalization, dimensionality reduction, and clustering are computationally intensive, and traditional tools often struggle to handle such datasets on commodity hardware. Furthermore, as the field of genomics advances, the atlas projects such as Human Cell Atlas[10] are expected to generate vast amounts of data, often measured in terabytes. To address this, a growing interest has emerged in the field of "big single-cell data science," which aims to adapt distributed computing tools to biological data[7]. The development of robust computational methods and big data tools to overcome these hurdles is essential for realizing the full potential of scRNA-seq in various biomedical applications, including cancer research, developmental biology, and personalized medicine[11]. Frameworks such as Apache Spark and Hadoop offer a robust foundation for scalable and fault-tolerant data processing, yet their integration into bioinformatics pipelines remains limited[12].

Despite the availability of powerful computational tools, there remains a lack of Spark-native frameworks specifically tailored to large-scale scRNA-seq analysis. Most existing pipelines, including popular tools such as Seurat[13] and Scanpy[14], rely on in-memory data structures and are constrained by available RAM, limiting their ability to scale to datasets in the range of hundreds of thousands to millions of cells.

In this work, we present scSPARKL, a modular, Spark-based analytical framework designed for scalable, parallel, and memory-efficient analysis of scRNA-seq data. The pipeline implements parallel routines for data reshaping, quality control, filtering, and normalization, while seamlessly integrating common downstream tasks such as dimensionality reduction and clustering. We demonstrate its effectiveness through the reanalysis of real-world datasets and benchmarking on varying dataset sizes to evaluate performance gains under different Spark configurations. To the best of our knowledge, this is the first attempt to leverage the computational power of Apache Spark and the efficient data storage capabilities of Apache Parquet to develop a framework specifically tailored for downstream analysis of large-scale scRNA-seq data, enabling seamless execution even on commodity hardware.

Related work

The rise of single-cell RNA sequencing (scRNA-seq) has sparked a boom in generating genomic data, bringing forth some complex computational challenges. Researchers have been exploring a variety of techniques and frameworks to deal with the vast scale and complexity of scRNA-seq datasets effectively. This breakthrough technology in genomics, single-cell RNA sequencing (scRNA-seq), has truly advanced up the field by allowing us to peek into gene expression at the individual cell level, triggering a wave of genomic data like never before[15]. With this surge in data volume comes a set of tough computational hurdles, thanks to the intricate nature of scRNA-seq datasets, packed with high dimensionality, sparsity, and noise[16]. To tackle these challenges head-on, researchers have proposed a number of techniques and frameworks, all aimed at efficiently handling the massive scale and complexity that comes with analyzing scRNA-seq data[17]. These strategies cover a wide range of computational methods, from slicing down dimensions with techniques like principal component analysis (PCA) and t-distributed stochastic neighbor embedding (t-SNE)[18], to clustering algorithms like k-means and hierarchical clustering and even diving into the realm of machine learning with approaches like deep learning for identifying cell types and inferring trajectories[19]. These methods are the key players in unraveling the intricate biological dances captured in scRNA-seq data, paving the way for a deeper dive into the world of cellular diversity and dynamics within the intricate web of biological systems.

The adoption of big data analytics tools like Apache Spark has significantly enhanced the processing and analysis of large-scale genomic datasets. Apache Spark's distributed computing capabilities have been instrumental in handling the vast amounts of genomic data efficiently[20]. Studies have showcased Spark's scalability, fault tolerance, and memory management advantages in genomic data analysis, making it a preferred choice for researchers[21]. In the realm of single-cell RNA sequencing (scRNA-seq) data analysis, Spark has been leveraged to create scalable pipelines and tools tailored to the unique challenges of analyzing individual cell transcriptomes.

Glow, a brainchild of the Regeneron Genetics Center and Databricks, shines as an open-source toolkit built on top of Apache Spark. It's a nifty tool designed to bring together genomic and phenotypic data using turbocharged algorithms for prepping genomic data, crunching numbers for statistical analysis, and diving into the world of machine learning on a grand scale akin to a biobank. This toolkit steps up to the plate, tackling the hurdles of handling hefty genomic datasets by offering a top-notch framework that speeds up the data engineering process for storing and analyzing genomic data on a grand scale. Glow acts as a bridge, connecting the dots between traditional bioinformatic toolkits and the modern data analytics landscape, paving the way for integrating machine learning into vast health datasets that encompass genomic data and a myriad of phenotypes[22].

While these existing frameworks and pipelines have been making great strides in tackling the computational hurdles of genomic data analysis, the rapid growth and complexity of datasets keep pushing the boundaries of scalability and performance. Moreover, the diverse range of experimental designs and analysis needs calls for the creation of adaptable and personalized solutions that cater to specific research objectives. Our proposed scSPARKL pipeline, aiming to introduce a fresh, scalable, and highly parallel computational framework for handling large-scale scRNA-seq data. By making use of the power of Apache Spark and bringing in innovative techniques for parallelization and optimization, scSPARKL aims to break through the constraints of current tools and enable the efficient analysis of massive scRNA-seq datasets using everyday hardware setups. This innovative approach underscores the ongoing commitment to crafting state-of-the-art solutions that can adeptly handle the intricacies of modern genomic data analysis, ultimately leading to deeper insights into cellular diversity and biological processes.

Methods

Dataset description

To evaluate the usability and versatility of the scSPARKL pipeline, we reanalyzed three publicly available single-cell RNA-seq datasets, each representing a distinct biological context and data structure:

Mouse brain non-myeloid cells

This dataset is part of the Tabula Muris Senis compendium, which profiles single-cell transcriptomes across 20 tissues from Mus musculus, comprising approximately 100,000 cells in total[23]. For our analysis, we extracted the subset of non-myeloid brain cells, which includes 3,401 cells with gene expression measurements for 23,433 genes. The accompanying metadata provides detailed annotations such as sub-tissue origin (cortex, hippocampus, cerebellum, and striatum), cell ontology classifications, mouse ID, and sex.

Jurkat-293T cell mixture

This dataset, originally published by[24], contains 3,388 cells derived from a 1:1 in vitro mixture of human 293 T and mouse Jurkat cell lines. The dataset was designed to distinguish between species-specific expression profiles. Cells expressing CD3D were classified as Jurkat, while those expressing XIST were identified as 293T. This dataset serves as a clear benchmark for evaluating interspecies separation and clustering accuracy.

68 K PBMC

To assess the pipeline's performance on larger datasets and evaluate runtime scalability, we used the 68 K Peripheral Blood Mononuclear Cell (PBMC) dataset, also from[24] The dataset consists of single-cell transcriptomic profiles collected from a healthy donor, encompassing a heterogeneous population of immune cells. This dataset was used to generate random subsamples of increasing sizes (2 K to 10 K cells) for benchmarking temporal performance.

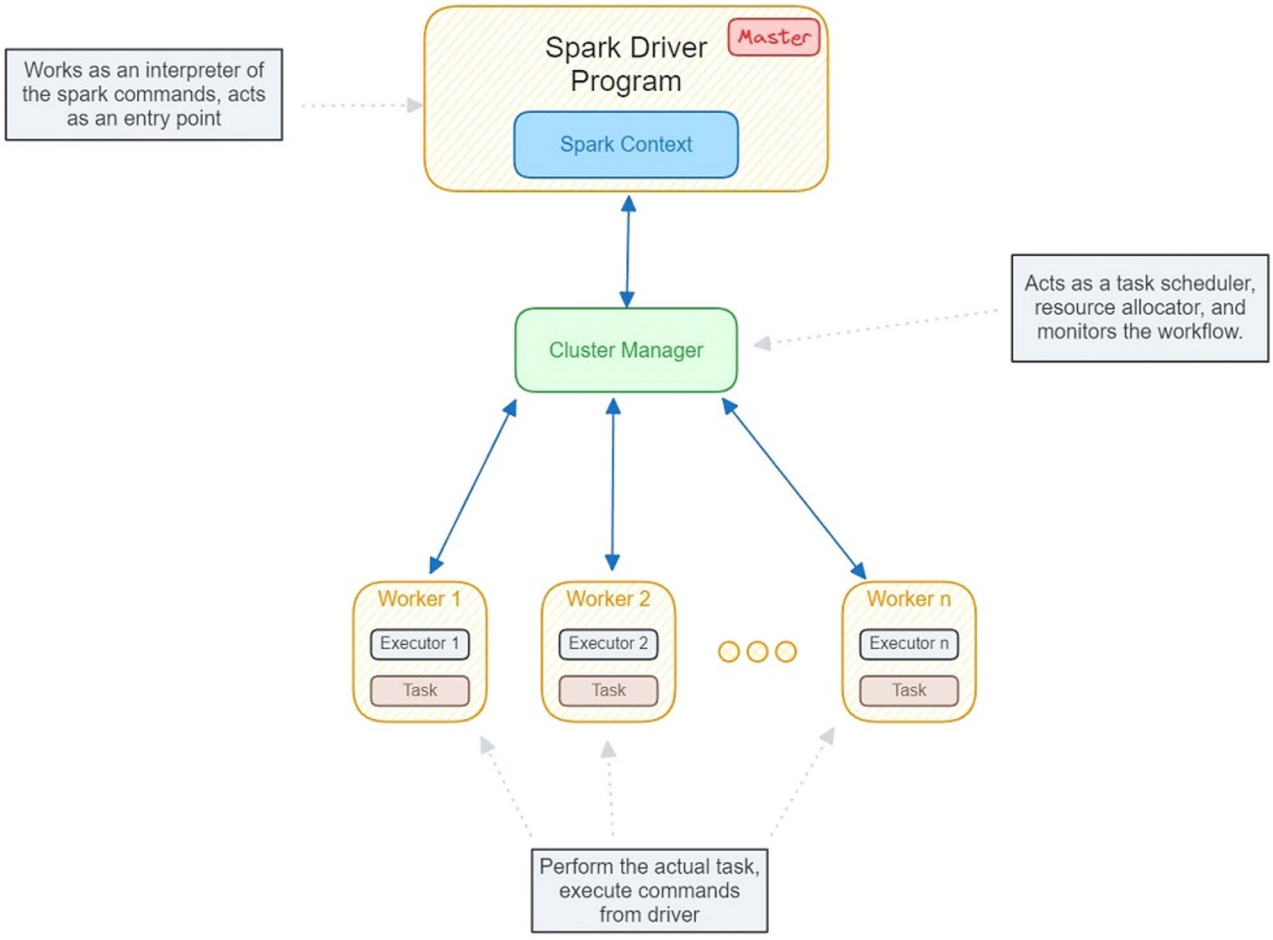
Pipeline development

For development, we used Apache's Spark version 3.1.2, Python 3.9 and JDK version 8.0. Apache Spark is a distributed analytical engine made for handling big data. It provides an essential parallel processing platform for large datasets[25]. Spark at its core uses an abstraction called Resilient Distributed Datasets (RDD), a read-only collection of objects distributed across the cluster of machines, which can be recreated if one of the machines is lost—fault tolerance[12]. All the jobs are launched in parallel through a program called "driver program" which also maintains the high-level flow control of the applications and jobs submitted to the executors (Fig. 1). Spark uses in-memory concept i.e., it leverages Random Access Memory (RAM) of a machine and brings the data to be processed directly to the RAM thereby increasing the response time by removing the time taken to fetch data from a disc drive[25].

The scSPARKL pipeline consists of 3 stages, algorithms for all the three stages (Pseudocode 1, 2, and 3) were designed in such way that a normal commodity hardware will support the analysis, and a possible optimization of spark is achieved. The stages are modular in nature supporting the variety of analysis. For a thorough understanding of how the stages work internally we have designed a Data Flow Diagram (DFD) for each of the stage (Supplementary Sect. 1).

Stage-1 preprocessing

Two exclusive Spark modules were developed for loading data and data filtering. The data_load.py package file loads the data as a Spark dataframe and performs the initial preprocessing like removal of special characters such as periods (.), commas (,), blank spaces etc., from the columns and replaces them with the underscore ('_'). The scRNA-seq data is usually in wide format and Spark has the ability to work better with tall/long format data; hence for optimization, we reshape the data to the tall format using *melt_Spark()* function in the pipeline, thereby reducing the columnar load and increasing efficiency. This step is essential for the users with low memory machines (i.e., as lows as 4GB). The dataframe is then written as a Spark Parquet file. Apache Parquet format is an open-source columnar storage file format which is based on "shredding and assembly algorithm". It is designed to handle large and complex datasets for faster aggregation functions. The parquet formatted file is compressed, which speeds up the further analytical procedures on the file. We mainly use Apache Sparks aggregate functions, known for their quick retrieval and less computationally exhaustive/expensive. Pseudocode-1 provides the detailed outline of the data cleaning and reshaping.



1: Apache Spark workflow

Fig. 1. Apache Spark workflow. The Spark driver program works as a master and as an entry point for all the Spark jobs. The master submits jobs to the worker nodes. The cluster manager keeps the track of the nodes, and the jobs distributed to them, several cluster managers are Yet Another Resource Negotiator (YARN), Kubernettes, mesos and standalone (in our case). The worker/slave nodes are the actual machines where the tasks are executed, and they report back to the cluster manager.

Pseudo Code 1: *Stage-1*

Require: expression count matrix in *cell* x *gene* format.
Output: Cleaned_Matrix, Melted_Matrix.

```
Input_Matrix →M_input [][],
Cleaned_Matrix →C_mat [][]
Clean_Spark_Dataframe→CSD, Spark_Melted_Data□SMD
START
 for Gene_name && Cell_name in M_input[][] do:
        C_mat= regexp_replace(M_input[][], special char to underscore)
 end for
 if (is_wide(C_mat[][])==TRUE)
        Tall_mat [][]= melt_to_tall(C_mat[][], value_vars, id)
        Tall_mat[][].repartition()
 end if
Tall_mat.persist()
Tall_mat.collect()  //actions
CSD=storeAsParquet (C_mat[][])
SMDF=storeAsParquet (Tall_mat[][])
Call stage 2
END
```

Stage-2 quality control

For the coherent and accurate downstream analysis of scRNA-seq data, it is necessary to filter out low-quality cells and genes. We followed "scater" package[9], SCANPY[14] and Seurat[13] for generating variety of quality control matrices. Quality matrix is calculated on number of parameters like total genes expressed per cell, percentage of external RNA controls consortium (ERCC), number cells expressing genes greater than zero, mitochondrial percentage detected in each cell etc. We also calculate the log measurements and mean measurements of numerous parameters for the easier interpretation such that a good parameter threshold for cell and gene filtering is defined.

If we have a cell/gene matrix $\boldsymbol{A_{ij}}$,

Where.

$i \in \{cell_1, cell_2, cell_3, \ldots, cell_n\}$ &.

$j \in \{gene_1, gene_2, gene_3, \ldots, gene_m\}$ then,

then Θ_1 and Θ_2 denote cell quality and gene quality summaries, respectively, can be given as following in the Tables 1 and 2.

In addition to the normal filtering procedures, we have implemented a parallelly executable Median Absolute

Cell Quality Measure	Symbol	Description	Formula
Total expression per cell	S_i	Sum of expression values for all genes in cell *i*	$S_i = \sum_{j=1}^{m} A_{ij}$
Number of genes expressed per cell	G_i	Count of genes with non-zero expression in cell *i*	$G_i = \sum_{j=1}^{m} 1_{(A_{ij}>0)}$
Average gene expression in a cell	AE_i	Mean expression across expressed genes in cell *i*	$AE_i = \frac{1}{G_i} \sum_{j:A_{ij}>0} A_{ij}$
Total ERCC expression	S_i^{ERCC}	Sum of all ERCC gene counts in cell *i*	$S_i^{ERCC} = \sum_{j \in ERCC} A_{ij}$
% ERCC expression	P_i^{ERCC}	Percentage of ERCC expression in total expression	$P_i^{ERCC} = \left(S_i^{ERCC} / S_i \right) \times 100$
Total mitochondrial expression	S_i^{MT}	Sum of all mitochondrial gene counts in cell *i*	$S_i^{MT} = \sum_{j \in MT} A_{ij}$
% Mitochondrial expression	P_i^{MT}	Percentage of mitochondrial expression	$P_i^{MT} = \left(S_i^{MT} / S_i \right) \times 100$
Log1p of total expression	$log1p\,(S_i)$	Log-transformed total expression per cell	$log1p\,(S_i) = \log(S_i + 1)$
Log1p of ERCC expression	$log1p\left(S_i^{ERCC}\right)$	Log-transformed ERCC counts	$log1p\left(S_i^{ERCC}\right) = \log(S_i^{ERCC} + 1)$
Log1p of mitochondrial expression	$log1p\left(S_i^{MT}\right)$	Log-transformed mitochondrial counts	$log1p\left(S_i^{MT}\right) = \log(S_i^{MT} + 1)$

Table 1. Θ_1 summaries and formulas for cell quality.

Gene Quality Measure	Symbol	Description	Formula
Number of cells expressing the gene	C_j	Count of cells in which gene *j* is expressed	$C_j = \sum_{i=1}^{n} 1_{(A_{ij}>0)}$
Total gene expression	T_j	Sum of expression values for gene *j*	$T_j = \sum_{i=1}^{n} A_{ij}$
Number of dropouts	D_j	Number of cells where gene *j* is not expressed	$D_j = n - C_j$
Dropout percentage	P_j^{drop}	Percentage of cells with zero expression	$P_j^{drop} = \left({D_j}/{n}\right) \times 100$
Mean expression in expressing cells	μ_j	Mean of non-zero expression values	$\mu_j = \frac{1}{C_j} \sum_{i:A_{ij}>0} A_{ij}$
Log1p of total gene expression	$log1p\,(T_j)$	Log-transformed total expression	$log1p\,(T_j) = log\,(T_j + 1)$
Log1p of mean expression	$log1p\left(\mu_j\right)$	Log-transformed mean expression	$log1p\left(\mu_j\right) = log\left(\mu_j + 1\right)$

Table 2. Θ_2 summaries and formulas for gene quality.

Deviation (MAD) filtering methodology for filtering out the genes with a normalized count value greater than a decided threshold (usually > 3). Pseudocode-2 outlines the overall procedure for computing cell and gene quality measurements, while pseudocode-3 outlines the filtering process based on these quality summaries along with subsequent steps of normalization, dimensionality reduction and clustering, which are described in detail in the following sections.

Pseudo Code 2: *Stage-2*

Require: cleaned tall format matrix CTM
Output: cell_quality_summary, gene_quality_summary.

Cleaned_tall_matrix → CTM, column_name → CN
Cell_quality_summary →CQS, Gene_quality_summary → GQS,
∀ ***Θ*** ∈ *cell summary operations*
∀ ***α*** ∈ *gene summary operations*
∀ ***action*** ∈ *{Spark.sum(), Spark.avg(), Spark.mean() etc.}*
START
for *each i ∈ CN in CQS* ***do:***
    CQS *=CQS.****Withcolumn****(CN, CTM.****Spark.sql.when****(Θ)*
end for
*CQS.****groupBy****(cell).****agg(Spark.action****(value, CQS)) //Spark action*
for *each new column name in GQS* ***do:***
    GQS *=CTM.****withColumn****(CN,* ***Spark.sql.when****(α) //transformation*
end for
*CQS.****groupBy****(cell).****agg(Spark.action****(key, value)) //Spark actions*
*GQS.****groupBy****(gene).****agg(Spark.action****(key, value)) //Spark actions*
*StoreAsCSV****CQS***
*StoreAsCSV****GQS***
Call stage-3
END

Stage 3 normalization and dimensionality reduction

Due to the inherent sparsity and variability of single-cell RNA-seq (scRNA-seq) data, normalization is a critical preprocessing step to ensure that downstream analyses reflect true biological signals rather than technical noise. Proper normalization improves the comparability of gene expression across cells and enhances the accuracy of clustering, dimensionality reduction, and differential expression analyses. The *data_normalize.py* module in the scSPARKL framework currently supports two normalization methods: Quantile Normalization (QN) and Global Normalization (GN).

In QN, the gene expression values across each cell is first ranked from lowest to highest, The data is arranged according to the ranks of the rows. Across all cells, values at the same rank are averaged to form a new reference distribution. These average values are then mapped back to each cell according to the original rank order. This method helps standardize expression distributions across cells while preserving relative expression relationships within each cell. It is particularly useful in making datasets with different expression scales more comparable.

Global normalization, on the other, adjusts each cell's total expression to a fixed scale (e.g., counts per 10,000), followed by a log-transformation. This approach corrects for differences in sequencing depth and is commonly used in standard scRNA-seq workflows. In this, we divide each expression value of a cell by the total number of counts of a gene. The resultant value is then multiplied by 10,000 and log transformed with the addition of a pseudo-count 1. Mathematically, if X_{n*m} is an unnormalized filtered scRNA-seq matrix, then the Normalized matrix X_N can be given as:

$$X_N = \left\{ \frac{x_{i,j}}{\sum_{i,j=0}^{i \leq n, j \leq m} (x_{i,j})} * 10{,}000 \right\} \tag{1}$$

$$where\ x\ \in\ X_{m*n}\ and\ 10{,}000\ is\ a\ scaling\ factor$$

and this scaled normalized matrix is then log transformed as:

$$X_{SN} = (loglog\ (X_N) + 1) \tag{2}$$

$$where\ 1\ is\ a\ pseudocount.$$

Both normalization strategies are integrated into the Spark-based pipeline to support parallel execution and efficient processing of large-scale datasets.

Dimensionality reduction

Dimensionality reduction is a key step in scRNA-seq analysis, aimed at projecting high-dimensional gene expression data into a lower-dimensional space while preserving the underlying structure and relationships among cells[26]. Due to the high dimensionality and sparsity of scRNA-seq data, conventional analysis methods can be impacted by the "curse of dimensionality"—a phenomenon where the distance between data points becomes less meaningful in high-dimensional spaces[27].

In this study, we employed three widely used dimensionality reduction techniques: Principal Component Analysis (PCA), t-distributed Stochastic Neighbor Embedding (t-SNE), and Uniform Manifold Approximation and Projection (UMAP). Among these, PCA is implemented using Spark's MLlib library, allowing distributed and parallel computation across nodes. This integration aligns with the design goals of scSPARKL to support scalable, high-throughput analysis.

By contrast, UMAP and t-SNE are currently implemented using standard Python libraries and operate outside the Spark execution engine. Nevertheless, these methods read input directly from intermediate Apache Parquet files, benefiting from efficient I/O and compatibility with Spark-generated outputs. While these steps are not fully parallelized within Spark, their integration into the pipeline remains seamless and supports downstream visualization and clustering workflows.

Clustering and further downstream analysis

A major advantage of single-cell RNA sequencing (scRNA-seq) lies in its ability to identify both known and novel cell populations through unsupervised clustering[28]. As single-cell profiling technologies continue to evolve, improvements in capture efficiency have enabled the sequencing of millions of individual cells, giving rise to increasingly large and complex datasets. This scale imposes substantial computational demands on clustering algorithms used for cell-type discovery and classification[29].

To address this, scSPARKL integrates the K-means clustering algorithm, which partitions a dataset of n observations into k clusters ($k \leq n$) by minimizing the within-cluster variance. The algorithm operates iteratively, first by randomly selecting k centroids and then by assigning each data point to the nearest centroid. This process continues until convergence, resulting in compact and well-separated clusters.

To support informed selection of k, the pipeline includes both the elbow plot method and silhouette score analysis, allowing users to evaluate clustering stability and separation. Once clusters are defined, differential gene expression (DGE) analysis is performed to identify genes that are most distinctively expressed between groups. For this, we apply a two-sample t-test across clusters and rank the top 10 significant genes based on their p-values, providing an interpretable set of marker genes for downstream biological validation.

Pseudo Code 3: *Stage-3*

Require: cleaned tall matrix, cell quality summary, gene quality summary.
Output: filtered matrix, normalized matrix.

```
Cleaned_tall_matrix → CTM,
Cell_quality_summary →CQS, Gene_quality_summary → GQS,
filtered_matrix →FM.
Vector_assembly → ρ
∀ μ ∈ normalization type
∀ ∂ ∈dimensionality reduction type
START
  if (filter_cells == TRUE) do:
      ercc_count = FM.contains('ercc').count()
      For each cell i in CQS, do:
          filtered_cells = mapPartition(filter_cells)
      End for
  end if
  if (filter_genes == TRUE) do:
      for each gene j in GQS do:
          filtered_genes = mapPartition(filter_genes)
      End for
  end if
  if (filter_cells = = TRUE) do:
      FM = CTM.inner_join(filtered_cells)
  end if
  if (filter_genes = = TRUE) do:
      FM = CTM.inner_join(filtered_genes)
  end if
  NM = normalize(FM, μ ).collect()
ρ = reduce_dimensions(∂ , NM).collect()
 apply_Kmeans(ρ, k)
 END
```

Results

The development and the execution of the pipeline was done on a Windows 11 OS, with 8GB RAM, intel i7 3.8Ghz processor with 8 cores. Table 3 provides the details of the hardware used in the experimentation. In Single-node Spark, one machine was used as a worker and the other one as a master machine. In the 2-node Spark environment, two machines were used as workers and one machine acted as a Master node. The memory allocation was done statically where each executor was provided with 2 GBs of memory. The rest of the memory is reserved for OS and Hadoop daemons.

The scSPARKL pipeline (Fig. 2) provides a promising approach for single-cell RNA sequencing (scRNA-seq) data analysis. To our knowledge it is the first Spark-based computational and analytical framework for scRNA-seq data, and leverages Spark's core engine to enable parallel resource utilization, dramatically reducing processing time. The pipeline efficiently performs a comprehensive suite of analyses, including data quality control, filtering (Fig. 3a, b), normalization, dimension reduction, clustering, and visualization, in less than 10 min. Our reanalysis results closely align with those presented by the original authors[23] (Brain Non-Myeloid, $n = 3401$) and[24] (PBMC, $n = 10000$), with total computation time as low as ~8 min, demonstrating both the efficiency and reliability of our approach. Importantly, the scSPARKL framework is fully open-source, providing the scientific community with a powerful tool that can be freely accessed, modified, and extended to meet evolving research needs. This open approach not only facilitates reproducibility but also encourages collaborative improvement and adaptation of the framework for diverse scRNA-seq applications.

Environment	Master/Slave platform	Cumulative RAM	No. of Cores	Processor type
Python sequential	×	6GiB	4	i7, 11th gen
Single-Node Spark	✓	6GiB	4	i7, 6th gen
2-Node Spark	✓	6 + 6 = 12GiB	8	i7, 5th gen and 6th gen

Table 3. Details of the hardware used in the experiments.

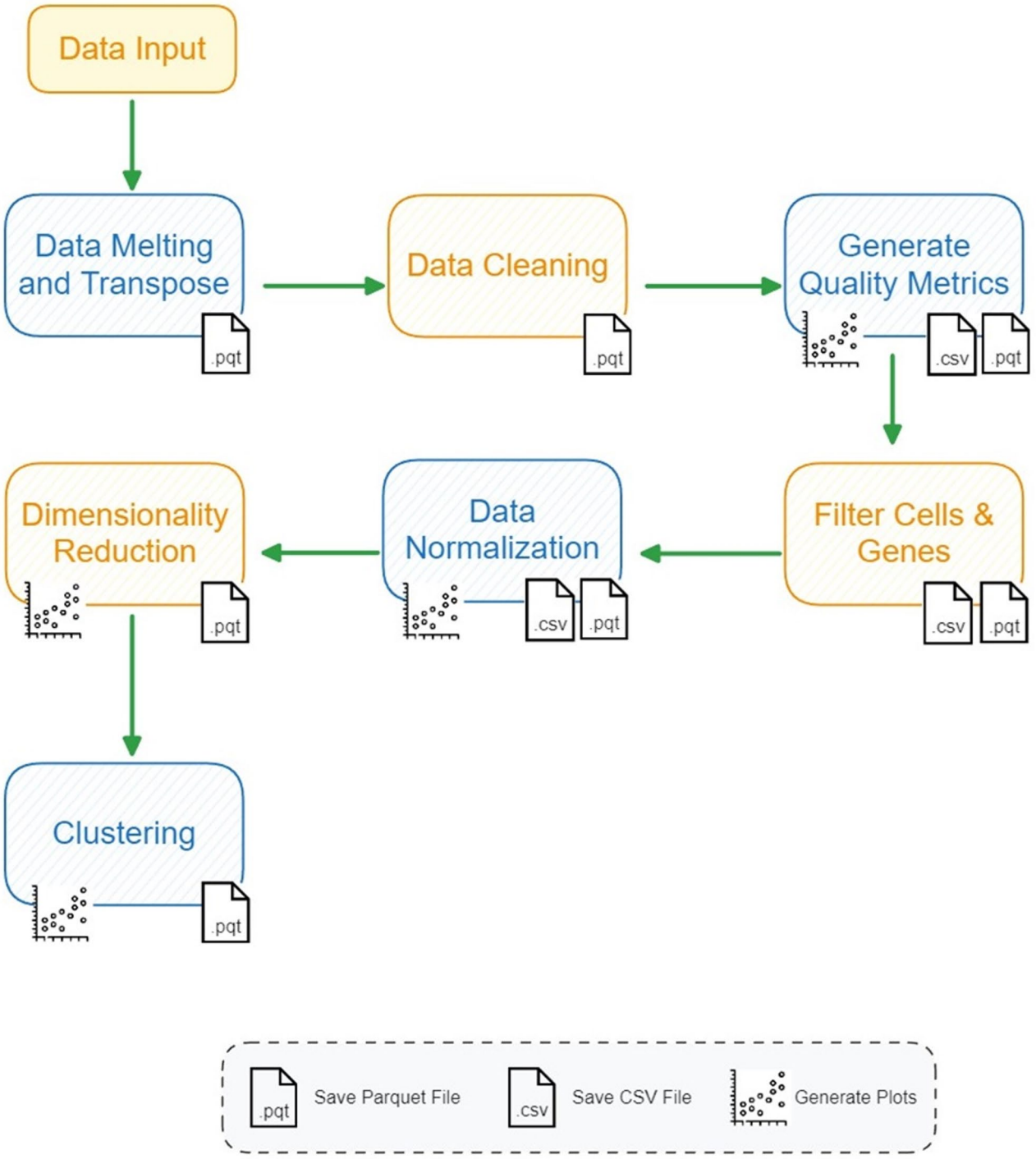


Fig. 2. General workflow of the scSPARKL pipeline. A complete, step-by-step description of each module is provided in the Materials and Methods section.

ScSPARKL shows consistency with prior analyses

To validate our framework, we performed reanalysis of Brain Non-myeloid cells and Jurkat-293T Cell data using scSPARKL. The outcomes of our analysis closely aligned with those of the original studies, demonstrating comparable biological accuracy. The framework accurately differentiates between cell types, yielding results consistent with those reported in the original studies. In the analysis of mouse brain non-myeloid cells, our pipeline identified a predominant presence of oligodendrocytes and endothelial cells, mirroring findings from

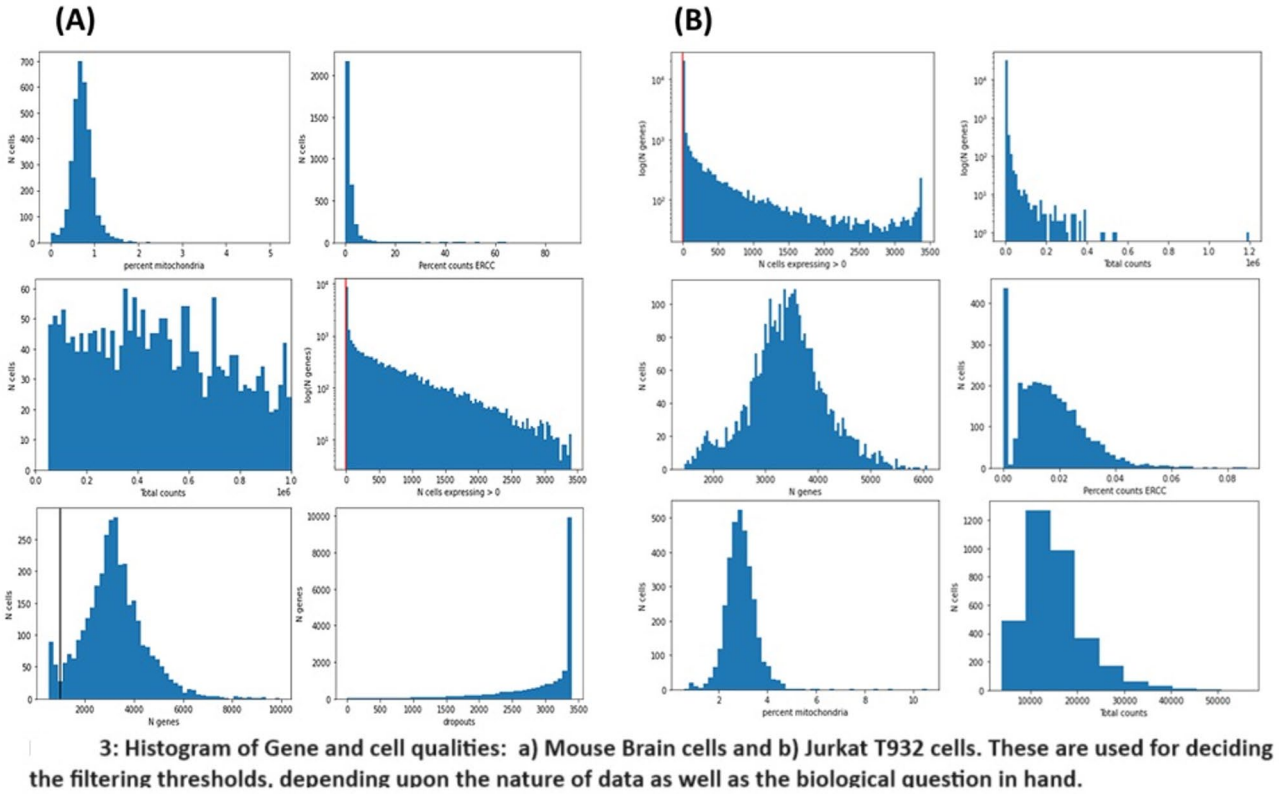


Fig. 3. Histogram of Gene and cell qualities. **(a)** Mouse Brain cells and **(b)** Jurkat T932 cells. These are used for deciding the filtering thresholds, depending upon the nature of data as well as the biological question in hand.

prior work(Fig. 4a, b). Similarly, in the Jurkat–293T cell mixture, scSPARKL clearly distinguished between mouse and human cells, effectively separating the two species in t-SNE space (Fig. 4c, d)., further validating the biological accuracy of the pipeline.

Some observed outliers in the t-SNE plots (Fig. 4d) likely reflect differences in cell and gene filtering thresholds as well as t-SNE parameter settings, which were kept at default for consistency across datasets. The default filter parameters used during analysis included: *percent_ercc* > *10%*, *percent_mito* > *5%*, *cells expressing* < *10 genes*, *genes expressed in* < *3 cells*, and *dropout rate* > *70%*. These conservative thresholds were chosen to ensure minimal manual intervention and to test the generalizability of the pipeline across datasets.

Additional UMAP plots supporting these findings are provided in the supplementary materials. Where appropriate, we followed the specific preprocessing and analysis protocols outlined in the original studies to enable direct biological comparison.

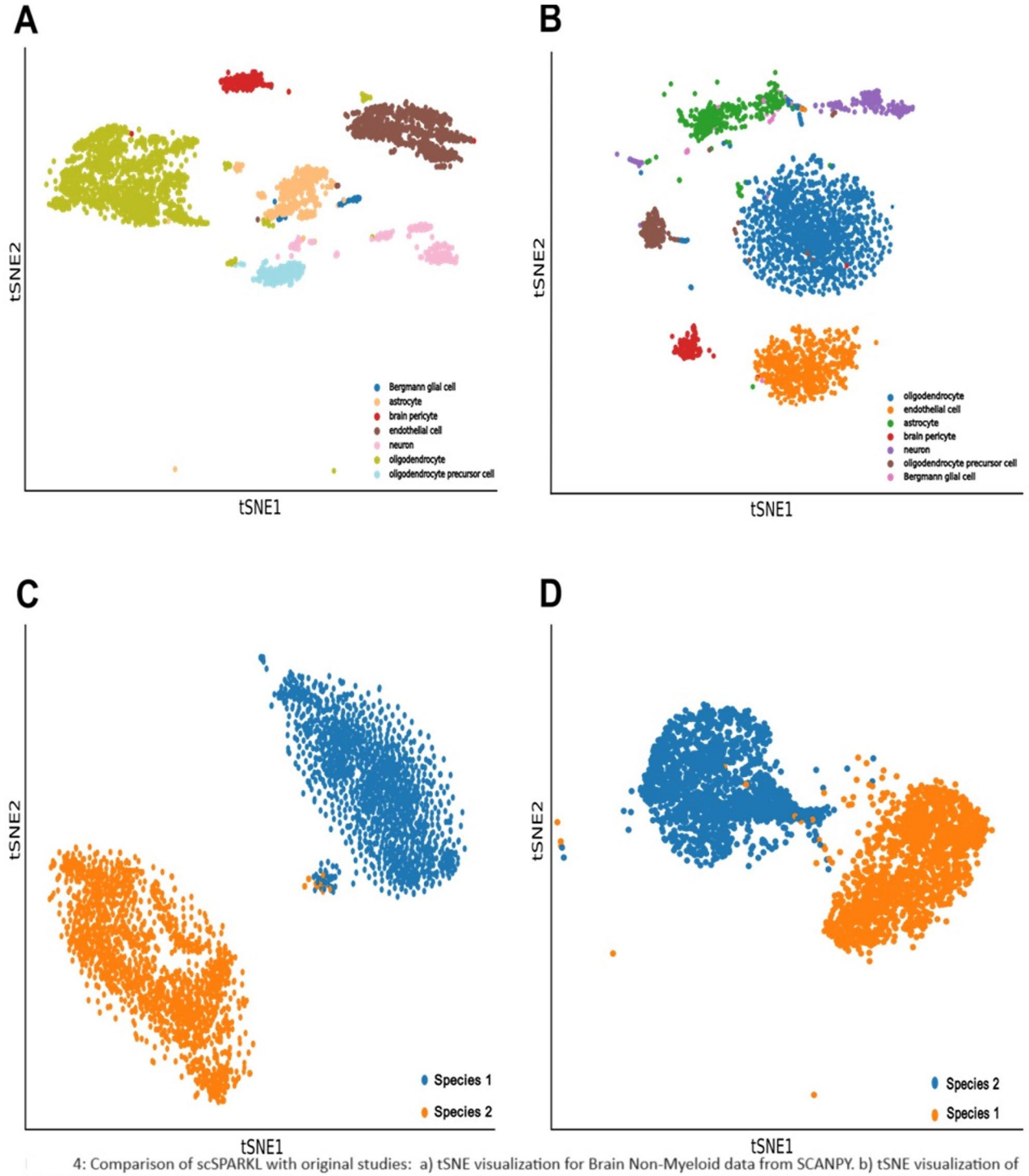
Clustering and concordance with annotated cell types

To assess the effectiveness of scSPARKL in identifying biologically meaningful groupings, we performed unsupervised clustering on the processed datasets. The number of clusters (k) was determined empirically for each dataset using a combination of the elbow method and silhouette analysis, which evaluate the variance explained and inter-cluster separation, respectively. These methods ensured that the chosen k-values aligned with both the underlying data structure and known biological annotations.

For quantitative validation of the clustering outcomes, we employed the Adjusted Rand Index (ARI)—a widely used metric for comparing predicted clusters with ground truth labels while correcting for chance. As shown (Fig. 5a, Supplementary Fig. S2–S5), the clustering results demonstrated varying degrees of concordance with annotated cell types across the datasets.

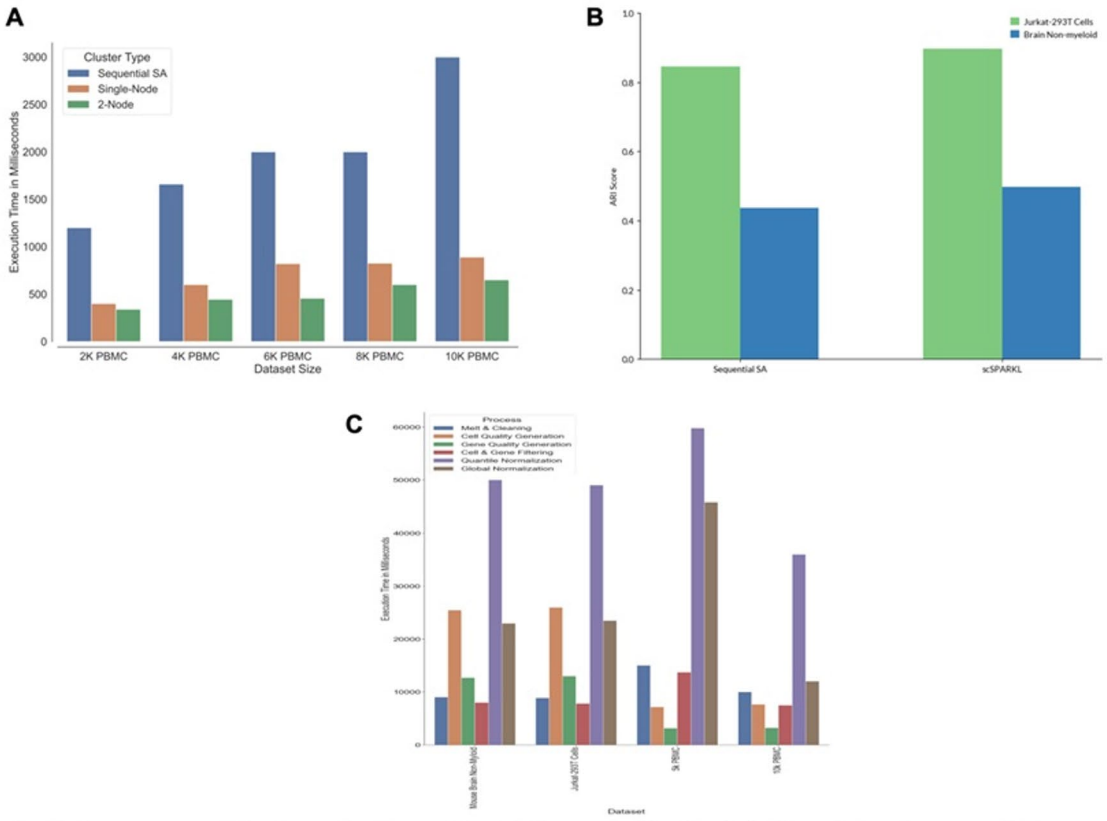
In the Jurkat–293T cell mixture, scSPARKL achieved an ARI of approximately 0.90, indicating strong agreement with the expected separation between human and mouse cells. This high score reflects the framework's ability to detect distinct transcriptomic signatures in datasets with clear species-specific boundaries. Additionally, for the brain non-myeloid dataset, the ARI was approximately 0.50, suggesting moderate agreement with annotated cell types. This result is consistent with the increased complexity and biological similarity among cell types in brain tissue, where subtle transcriptional differences can make cluster resolution more challenging.

These results support the utility of scSPARKL for unsupervised cell-type identification and demonstrate its capacity to produce clustering outputs that are in line with established biological expectations.



4: Comparison of scSPARKL with original studies: a) tSNE visualization for Brain Non-Myeloid data from SCANPY. b) tSNE visualization of Brain Non-Myeloid cells using scSPARKL. c) tSNE visualization for Jurkat Cell data from original author. d) tSNE visualization of Jurkat-239T Cells using scSPARKL.

Fig. 4. Comparison of scSPARKL with original studies. **(a)** tSNE projection for Brain Non-Myeloid data processed with SCANPY. **(b)** Corresponding tSNE visualization of Brain Non-Myeloid cells generated with scSPARKL. **(c)** Reference tSNE visualization for Jurkat Cell data from original author. **(d)** tSNE visualization of Jurkat-239T Cells using scSPARKL. Legends are shared for panels (**a**) and (**b**), and separately for panels (**c**) and (**d**).



5: Performance and Cluster evaluation: a) Bar Plot representing the total time taken (in seconds) from submitting the job to the final output i.e., clustering, for different data with different cell loads. Cells were randomly sampled and chunked from 68K-PBMC data. b) ARI scores of Clustering results from Sequential Standalone scRNA pipeline and scSPARK. c) Bar Plot depicting the time (in Milliseconds) taken by each process on different data sizes. All the processes shown in the plot are exclusively Spark based, running parallelly on each core.

Fig. 5. Performance and Cluster evaluation. **(a)** Bar Plot representing the total time taken (in seconds) from submitting the job to the final output i.e., clustering, for different data with different cell loads. Cells were randomly sampled and chunked from 68 K-PBMC data. **(b)** ARI scores of Clustering results from Sequential Standalone scRNA pipeline and scSPARK. **(c)** Bar Plot depicting the time (in Milliseconds) taken by each process on different data sizes. All the processes shown in the plot are exclusively Spark based, running parallelly on each core.

Apache spark and parquet improve execution and performance

We utilized the Peripheral Blood Mononuclear Cell (PBMC) dataset to evaluate the performance and scalability trends of the scSPARKL pipeline. To systematically assess runtime behavior across increasing data volumes, we generated five randomly sampled subsets containing 2,000, 4,000, 6,000, 8,000, and 10,000 cells, each comprising over 30,000 genes.

Performance comparisons were conducted between scSPARKL and a sequential Python-based standalone implementation (SSA) that replicated similar preprocessing and analysis functionalities. Benchmarking was performed under three different computational environments: the SSA, a single-node Spark setup, and a two-node Spark cluster. For the largest tested dataset (10,000 cells), scSPARKL processed the data in approximately 10 min on the two-node cluster and 13 min on the single-node Spark environment (Fig. 5b, c), compared to over 50 min required by the SSA. This corresponds to observed runtime improvements of approximately 80% and 74% for the two-node and single-node configurations, respectively.

Although the evaluation was limited to mid-sized datasets, the linear scalability trends observed across these experiments are consistent with the expected behavior of Spark-based distributed processing. These results suggest that scSPARKL is well-positioned to handle larger datasets as computational resources are expanded. We anticipate that increasing physical cores and RAM could potentially yield speed improvements exceeding 100x.

Performance analysis

To evaluate the computational efficiency and scalability of scSPARKL, we conducted a theoretical analysis of the time complexity associated with each major component of the pipeline. These components include data reshaping, quality control metric calculations, normalization (both quantile and global), dimensionality reduction through PCA, clustering using k-means, and differential gene expression analysis. In a sequential

environment, these operations often present significant computational bottlenecks, particularly as the number of cells (N) and genes (G) grows. However, by leveraging Apache Spark's distributed computing capabilities, we are able to parallelize these tasks and distribute the load across multiple executor cores, thereby significantly reducing the overall execution time.

Each operation was modeled mathematically to estimate its time complexity in both sequential and parallelized Spark-based implementations. For instance, data reshaping and quality control scale as $\mathcal{O}(NG)$ in sequential form, but are optimized to $\mathcal{O}(NG/P)$when distributed across P Spark partitions. More computationally intensive steps, such as quantile normalization and PCA, benefit even more from Spark's in-memory execution and lazy evaluation model. Collectively, the total computational complexity of the pipeline under Spark optimization is expressed as:

$$T_{total} = O\left(\frac{NG+NGlogG+Nk^2+Nkt}{P} + k^3\right) \tag{3}$$

where N is the number of cells, G the number of genes, k the number of principal components or clusters, T the number of clustering iterations, C the number of clusters used in DGE, and P the number of Spark executors. This equation reflects the direct benefit of parallelism and the linear reduction in processing cost as executor resources increase. It is also indicative of the modular nature of scSPARKL—each stage independently benefits from parallel execution without introducing additional overhead from interdependencies. A comprehensive breakdown of these derivations, along with a log–log scaling visualization, is provided in **Supplementary File S1**.

Our analysis revealed several optimization strategies:

1. Data Persistence: Utilizing the *persist()* function in Apache Spark to cache dataframes improved execution time for processes such as generating quality summaries and filtering low-quality cells and genes. However, this approach consumes significant executor RAM.
2. Data Storage Format: Employing the Parquet format for data storage proved advantageous, reducing shuffle-read time.
3. Spark API Utilization: We leveraged the Spark DataFrame API and SQL functions instead of Resilient Distributed Datasets (RDDs). This approach mitigated issues related to Java serialization and garbage collection, which can increase memory overhead in low-memory scenarios.

Implementation and deployment

scSPARKL is implemented in Python using PySpark and is designed to be lightweight, modular, and easy to deploy on a wide range of systems. The framework can be executed on:

- a standalone machine using a local Spark session, or.
- a distributed multi-node Spark cluster for large-scale datasets.

To ensure accessibility for users, we provide clear instructions and environment configuration scripts for setting up scSPARKL on machines with as little as 4 GB RAM and a 4-core processor. The source code, complete installation instructions, and usage examples are publicly hosted on GitHub: https://www.github.com/asif7adil/scSPARKL. This design philosophy ensures that scSPARKL is easy to install, scalable, and extensible, catering to a broad range of users from data scientists to biologists with minimal programming expertise.

Discussion

The exponential growth of single-cell transcriptomic data has created an urgent need for efficient tools capable of handling this data explosion. scSPARKL, was developed to address the Big Data characteristics of single-cell data, offering a robust solution for processing and analyzing large-scale scRNA-seq datasets. By implementing Apache Spark as the core engine, scSPARKL ensures unbounded parallelism, fault tolerance, and rapid computation of vast amounts of data. This approach sets our tool apart from established alternatives like Scanpy[14], which relies on Pandas for data handling. The limitations of Pandas in processing terabytes of data and its lack of parallelization capabilities highlight the novelty and scalability of our Spark-based approach.

Our pipeline incorporates several innovative features in its architecture and preprocessing stages. The framework's ability to handle both tall and wide format data, coupled with efficient data cleaning and quality check processes, provides a flexible and user-friendly approach to scRNA-seq data analysis. The generation of separate quality dataframes for cells and genes, stored as parquet files, facilitates mid-pipeline analytical entry points and eliminates the need for repeated data melting. Additionally, incorporation of parquet as a storage option further enhances the computational process. This indicates the usage of this storage format can be essential in near future for efficient querying of scRNA-seq data. Furthermore, scSPARKL offers robust, parallel normalization methods and flexible gene selection approaches. These features allow users to tailor the analysis to their specific research needs while maintaining computational efficiency. The implementation of parallel versions of dimensionality reduction techniques, including PCA, UMAP, and t-SNE, further enhances the pipeline's versatility and performance.

While large-scale testing remains ongoing, the scSPARKL framework demonstrates strong potential for efficient and scalable scRNA-seq data analysis. Its current performance on mid-sized datasets suggests it could be a practical solution for handling larger single-cell transcriptomic datasets as computational resources allow. Additionally, the open-source nature of scSPARKL also encourages community-driven improvements and adaptations. However, a potential limitation of our framework is the substantial RAM requirement in standalone

environments, particularly for post-normalization steps that necessitate wide format data. This challenge is currently mitigated through data melting techniques, but further optimization is needed.

The present study primarily focuses on establishing the computational architecture, scalability, and modular workflow of scSPARKL, we acknowledge the importance of benchmarking new tools against widely adopted frameworks such as Seurat and Scanpy. These tools have become the gold standard for single-cell transcriptomics analysis, particularly for their comprehensive biological functionality and integration with well-established statistical workflows. However, it is important to highlight that scSPARKL was designed with a distinct emphasis on handling Big Data scale scRNA-seq experiments using distributed computing paradigms—targeting scenarios where traditional tools face memory bottlenecks or scalability limitations.

Because of the fundamental differences in infrastructure and optimization goals, a direct feature-by-feature comparison may not yield a fair or meaningful evaluation without aligning runtime environments, data volume, and hardware specifications. Nevertheless, we recognize that an evaluation of computational trade-offs — including runtime efficiency, memory utilization, and biological concordance — across multiple platforms would be valuable for end-users in choosing the appropriate tool for their datasets and infrastructure.

Although Apache Spark 3.0 supports GPU acceleration via RAPIDS-AI, the current version of scSPARKL does not utilize this functionality. This decision was made to ensure compatibility with standard computing environments, including academic and personal workstations. Our focus for this version was on maximizing performance through CPU-based parallelism, data storage optimization, and efficient use of Spark's native APIs. Future releases of scSPARKL will explore optional GPU acceleration modules, and performance benchmarks for those configurations will be made available accordingly. Furthermore, the future development will focus on optimizing Spark cluster mode operations, where each node has separate memory allocation, this promises to significantly increase processing speed and parallelism. As single-cell studies expand to include proteomic and genomic levels (e.g., CITE-seq, scATAC-seq, and scBS-seq for single-cell methylation), we plan to develop Spark-based pipelines for the integration of single-cell multi-omics data. This expansion will broaden the applicability of scSPARKL across various biological and medical research domains.

In conclusion, scSPARKL represents a significant step forward in the analysis of single-cell transcriptomic data. By leveraging the power of Apache Spark, our framework offers a scalable, efficient, and accurate solution for processing large-scale datasets. As the field of single-cell genomics continues to evolve, scSPARKL is well-positioned to adapt and grow, supporting researchers in unlocking new insights from increasingly complex and voluminous single-cell data.

Conclusion

Considering the unprecedented accumulation of scRNA-seq data over the past few years, we have developed a parallel computational framework that offers an intuitive and user-friendly approach to data management and subsequent analysis procedures. Our framework is a viable alternative to existing tools that lack scalability and fault tolerance, since the majority rely on pandas, which can collapse and stutter as data size exceeds 50GBs. By utilising Apache Spark at its foundation, we ensure that our framework can process terabytes of data at a breakneck pace. With the advent of single-cell studies at the proteomic and genomic levels (e.g., CITE-seq, scATAC-seq, and scBS-seq for single-cell methylation), we plan to develop Spark-based pipelines for the integration of single-cell multi-omics data.

Major findings

Our framework represents a major advancement in the analysis of single-cell transcriptomic data, with broad applications in the disciplines of biology and medicine. The implementation of Apache parquet format in the pipeline gave an additional boost up to the analysis, suggesting that Apache Parquet can be a good alternative to other file formats in the field. Additionally, an interesting finding and a possible hypothesis for the future is that the Apache spark shows a contrasting improvement in execution time when the dataset size is increased, indicating that the single-cell data accumulation, on a multi-omics scale, can be managed by developing the big data-based pipelines.

Data availability

The framework implementation and the source code, using PySpark, can be found on following: https://github.com/asif7adil/scSPARKL.

Received: 16 March 2025; Accepted: 21 July 2025
Published online: 29 July 2025

References

1. Wu, A. R., Wang, J., Streets, A. M. & Huang, Y. Single-Cell transcriptional analysis. *Annual Rev. Anal. Chem.* **10** (1), 439–462. https://doi.org/10.1146/annurev-anchem-061516-045228 (2017).
2. Jindal, A., Gupta, P. & Sengupta, D. Discovery of rare cells from voluminous single cell expression data. *Nat. Commun.* https://doi.org/10.1038/s41467-018-07234-6 (2018).
3. Slovin, S. et al. Single-Cell RNA sequencing analysis: A Step-by-Step overview. *Methods Mol. Biology (Clifton N J)*. **2284**, 343–365. https://doi.org/10.1007/978-1-0716-1307-8_19 (2021).
4. Vallejos, C. A., Marioni, J. C. & Richardson, S. BASiCS: Bayesian analysis of single-cell sequencing data. *PLoS Comput. Biol.* https://doi.org/10.1371/journal.pcbi.1004333 (2015).
5. Wen, L. et al. Single-cell technologies: From research to application. *Innovation* https://doi.org/10.1016/J.XINN.2022.100342 (2022).
6. Bhattacharya, N., Nelson, C. C., Ahuja, G. & Sengupta, D. Big data analytics in single-cell transcriptomics: Five grand opportunities. *Wiley Interdiscip. Rev.: Data Min. Knowl. Discov.* **11**(4), e1414. https://doi.org/10.1002/WIDM.1414 (2021).

7. Adil, A., Kumar, V., Jan, A. T. & Asger, M. Single-Cell transcriptomics: current methods and challenges in data acquisition and analysis. *Front. NeuroSci.* **15**, 398. https://doi.org/10.3389/FNINS.2021.591122/BIBTEX (2021).
8. Dhasmana, A. et al. Integrative big transcriptomics data analysis implicates crucial role of MUC13 in pancreatic cancer. *Comput. Struct. Biotechnol. J.* **21**, 2845–2857. https://doi.org/10.1016/J.CSBJ.2023.04.029 (2023).
9. McCarthy, D. J., Campbell, K. R., Lun, A. T. L., Wills, Q. F. & Bioinformatics Scater: pre-processing, quality control, normalization and visualization of single-cell RNA-seq data in R. 33(8), 1179–1186. (2017). https://doi.org/10.1093/BIOINFORMATICS/BTW777
10. Regev, A. et al. Science forum: The human cell atlas. *ELife* **6**, e27041 (2017).
11. Hassan, M. et al. Innovations in genomics and big data analytics for personalized medicine and health care: A review. *Int. J. Mol. Sci.* https://doi.org/10.3390/IJMS23094645 (2022).
12. Salloum, S., Dautov, R., Chen, X., Peng, P. X. & Huang, J. Z. Big data analytics on Apache spark. *Int. J. Data Sci. Analytics*. **1** (3–4), 145–164. https://doi.org/10.1007/S41060-016-0027-9/FIGURES/6 (2016).
13. Satija, R., Farrell, J. A., Gennert, D., Schier, A. F. & Regev, A. Spatial reconstruction of single-cell gene expression data. *Nat. Biotechnol.* **33** (5), 495–502. https://doi.org/10.1038/nbt.3192 (2015).
14. Wolf, F. A., Angerer, P. & Theis, F. J. SCANPY: Large-scale single-cell gene expression data analysis. *Genome Biol.* **19**(1), 1–5. https://doi.org/10.1186/s13059-017-1382-0 (2018).
15. Tang, F. et al. mRNA-seq whole-transcriptome analysis of a single cell. *Nat. Methods*. **6** (5), 377–382. https://doi.org/10.1038/nmeth.1315 (2009).
16. Stegle, O., Teichmann, S. A. & Marioni, J. C. Computational and analytical challenges in single-cell transcriptomics. *Nat. Reviews Genet. 2015*. **16:3** (3), 133–145. https://doi.org/10.1038/nrg3833 (2015).
17. Haque, A., Engel, J., Teichmann, S. A. & Lönnberg, T. A practical guide to single-cell RNA-sequencing for biomedical research and clinical applications. *Genome Med.* **9** (1), 75. https://doi.org/10.1186/s13073-017-0467-4 (2017).
18. Amir, E. A. D. et al. ViSNE enables visualization of high dimensional single-cell data and reveals phenotypic heterogeneity of leukemia. *Nat. Biotechnol. 2013*. **31** (6), 6. https://doi.org/10.1038/nbt.2594 (2013).
19. Angermueller, C. et al. Parallel single-cell sequencing links transcriptional and epigenetic heterogeneity. *Nat. Methods 2016*. **13** (3), 3. https://doi.org/10.1038/nmeth.3728 (2016).
20. Zaharia, M. et al. Apache spark. *Commun. ACM*. **59** (11), 56–65. https://doi.org/10.1145/2934664 (2016).
21. Guo, X., Yu, N., Ding, X., Wang, J. & Pan, Y. DIME: A novel framework for de Novo metagenomic sequence assembly. *Https://Home Liebertpub Com/Cmb*. **22** (2), 159–177. https://doi.org/10.1089/CMB.2014.0251 (2015).
22. Glow. (2019). https://projectglow.io/
23. Schaum, N. et al. Single-cell transcriptomics of 20 mouse organs creates a Tabula muris. *Nat. 2018*. **562** (7727), 367–372. https://doi.org/10.1038/s41586-018-0590-4 (2018).
24. Zheng, G. X. Y. et al. Massively parallel digital transcriptional profiling of single cells. *Nat. Commun.* **8**, 1–12. https://doi.org/10.1038/ncomms14049 (2017).
25. Aziz, K., Zaidouni, D. & Bellafkih, M. Leveraging resource management for efficient performance of Apache spark. *J. Big Data*. https://doi.org/10.1186/s40537-019-0240-1 (2019).
26. Ayesha, S., Hanif, M. K. & Talib, R. Overview and comparative study of dimensionality reduction techniques for high dimensional data. *Inform. Fusion*. **59**, 44–58. https://doi.org/10.1016/J.INFFUS.2020.01.005 (2020).
27. Feng, C. et al. Dimension reduction and clustering models for single-cell RNA sequencing data: A comparative study. *Int. J. Mol. Sci.* https://doi.org/10.3390/IJMS21062181 (2020).
28. Menon, V. Clustering single cells: a review of approaches on high-and low-depth single-cell RNA-seq data. *Brief. Funct. Genom.* **18**(6), 434. https://doi.org/10.1093/bfgp/ely001 (2018).
29. Baker, D. N., Dyjack, N., Braverman, V., Hicks, S. C., & Langmead, B. (2021). Fast and memory-efficient scRNA-seq k-means clustering with various distances. ACM-BCB… The … ACM Conference on Bioinformatics, Computational Biology and Biomedicine. ACM Conference on Bioinformatics, Computational Biology and Biomedicine, 2021, 24, p1-8.https://doi.org/10.1145/3459930.3469523

Acknowledgements
The authors are highly thankful to Prof. Debarka Sengupta (Computational Biology division of IIIT-D, India), for his valuable inputs and suggestions which improved the overall quality of the manuscript.

Author contributions
A.A, and N.J.K conceived the study. A.A developed the package under the supervision of M.A. A.A and N.J.K performed analysis. A.A, N.B, A., and N.J.K helped in manuscript writing and incorporating revisions. All authors discussed the results, co-wrote and reviewed the manuscript.

Funding
No funding was availed for the study.

Declarations

Competing interests
The authors declare no competing interests.

Additional information
Supplementary Information The online version contains supplementary material available at https://doi.org/10.1038/s41598-025-12897-5.

Correspondence and requests for materials should be addressed to A.A., N.J.K. or M.A.