GigaScience

Transcriptome annotation in the cloud: complexity, best practices and cost.

--Manuscript Draft--

Manuscript Number:	GIGA-D-20-00202
Full Title:	Transcriptome annotation in the cloud: complexity, best practices and cost.
Article Type:	Technical Note
Funding Information:	U.S. National Library of Medicine (Intramural Research Program of the National Library of Medicine, National Center for Biotechnology Information at the National Institutes of Health.) — Dr. David Landsman
Abstract:	Background Transcriptome annotation is a complex analytical process that requires the integration of multiple biological databases and several advanced computational tools. The core components of annotation pipelines published since 2012 are BLAST jobs using annotated databases of both nucleotide or protein sequences almost exclusively with networked on premises compute systems. Public cloud compute providers represent an alternative for the execution of large computational biology experiments like BLAST alignments, yet little has been published describing cloud computing implementation best practices and cost estimates. Findings We present a comparative study of multiple BLAST sequence alignments using two public cloud providers: Amazon Web Services (AWS; Seattle, WA, USA) and Google Cloud Platform (GCP; Mountain View, CA, USA). We have prepared several Jupyter Notebooks with all the code required to submit BLAST jobs to the batch system on each cloud provider. We consider the consequence of the number of query transcripts in an input file and the effect on cost and processing time. We tested compute instances with 16, 32 and 64 vCPUs on each cloud provider. Four classes of timing results were collected: the total running time, the time for transferring the BLAST databases to the instance local solid state disk drive (SSD), the time to execute the Common Workflow Language (CWL) script and the time for the creation, setup and release of an instance. This study aims to establish an estimate of the cost and compute time needed for the execution of multiple BLAST runs in a cloud environment. Conclusions We demonstrate that the public cloud providers are a practical alternative for executing advanced computational biology experiments at quite low cost. Using our cloud recipes, the BLAST alignments required to annotate a transcriptome with ~500,000 transcripts can be processed in less than 2 hours with a computing cost of about US$ 200-250. In our opinion, the choice of cloud platform is not dependent on the workflow but, rather, on the specific details and requirements of the cloud provider. These include the accessibility for institutional use of the cloud platforms, the technical knowledge required for effective use of the platform services, and the availability of open-source frameworks such as application programming interfaces (APIs) to deploy the workflow.
Corresponding Author:	David Landsman, Ph.D National Center for Biotechnology Information Bethesda, Maryland UNITED STATES
Corresponding Author Secondary Information:	
Corresponding Author's Institution:	National Center for Biotechnology Information
Corresponding Author's Secondary Institution:	

Powered by Editorial Manager® and ProduXion Manager® from Aries Systems Corporation

First Author:	Roberto Vera Alvarez, Ph.D
First Author Secondary Information:	
Order of Authors:	Roberto Vera Alvarez, Ph.D
	Leonardo Mariño-Ramírez, Ph.D
	David Landsman, Ph.D
Order of Authors Secondary Information:	
Additional Information:	
Question	**Response**
Are you submitting this manuscript to a special series or article collection?	No
Experimental design and statistics Full details of the experimental design and statistical methods used should be given in the Methods section, as detailed in our Minimum Standards Reporting Checklist. Information essential to interpreting the data presented should be made available in the figure legends. Have you included all the information requested in your manuscript?	Yes
Resources A description of all resources used, including antibodies, cell lines, animals and software tools, with enough information to allow them to be uniquely identified, should be included in the Methods section. Authors are strongly encouraged to cite Research Resource Identifiers (RRIDs) for antibodies, model organisms and tools, where possible. Have you included the information requested as detailed in our Minimum Standards Reporting Checklist?	Yes
Availability of data and materials All datasets and code on which the conclusions of the paper rely must be	Yes

either included in your submission or deposited in publicly available repositories (where available and ethically appropriate), referencing such data using a unique identifier in the references and in the “Availability of Data and Materials” section of your manuscript. Have you have met the above requirement as detailed in our Minimum Standards Reporting Checklist?	

Main Manuscript Click here to access/download;Manuscript;Main manuscript.docx

Transcriptome annotation in the cloud: complexity, best practices and cost.

Roberto Vera Alvarez[1], Leonardo Mariño-Ramírez[1,2], and David Landsman[1,*]

[1] Computational Biology Branch, National Center for Biotechnology Information, National Library of Medicine, NIH, Bethesda, MD, USA.

[2] Current address: Division of Intramural Research, National Institute on Minority Health and Health Disparities, NIH, Bethesda, MD, USA.

*To whom correspondence should be addressed.

Abstract

Background

Transcriptome annotation is a complex analytical process that requires the integration of multiple biological databases and several advanced computational tools. The core components of annotation pipelines published since 2012 are BLAST jobs using annotated databases of both nucleotide or protein sequences almost exclusively with networked on premises compute systems. Public cloud compute providers represent an alternative for the execution of large computational biology experiments like BLAST alignments, yet little has been published describing cloud computing implementation best practices and cost estimates.

Findings

We present a comparative study of multiple BLAST sequence alignments using two public cloud providers: Amazon Web Services (AWS; Seattle, WA, USA) and Google Cloud Platform (GCP; Mountain View, CA, USA). We have prepared several Jupyter Notebooks with all the code required to submit BLAST jobs to the batch system on each cloud provider. We consider the consequence of the number of query transcripts in an input file and the effect on cost and processing time. We tested compute instances with 16, 32 and 64 vCPUs on each cloud provider. Four classes of timing results were collected: the total running time, the time for transferring the BLAST databases to the instance local solid state disk drive (SSD), the time to execute the Common Workflow Language (CWL) script and the time for the creation, setup and release of an instance. This study aims to establish an estimate of the cost and compute time needed for the execution of multiple BLAST runs in a cloud environment.

Conclusions

We demonstrate that the public cloud providers are a practical alternative for executing advanced computational biology experiments at quite low cost. Using our cloud recipes, the BLAST alignments required to annotate a transcriptome with ~500,000 transcripts can be processed in less than 2 hours with a computing cost of about US$ 200-250. In our opinion, the choice of cloud platform is not dependent on the workflow but, rather, on the specific details and requirements of the cloud provider. These include the accessibility for institutional use of the cloud platforms, the technical knowledge required for effective use of the platform services, and

the availability of open-source frameworks such as application programming interfaces (APIs) to deploy the workflow.

Background

The annotation of RNA transcripts with functional and biological processes is an important step in developing an understanding of the biological complexity of an organism. In addition, annotation is a challenging process that requires the integration of multiple biological databases and several computational tools to accurately assign a function to an RNA product. Available public information on a target organism is the main limitation of the annotation of non-model organisms. The National Center for Biotechnology Information (NCBI) Genome database, for instance, contains 54,049 genome-sequencing projects by organism [1]. This includes 12,204 eukaryotes genomes for more than 1,000 species or strains at different assembly levels (95 complete genomes, 1,872 chromosomes, 7,743 scaffolds, and 2,494 contigs (https://www.ncbi.nlm.nih.gov/genome/browse/#!/eukaryotes/), accessed on June 30, 2020. Although these data include an important group of organisms, there is a lack of annotation of several species that have significant public health and economic importance. Significantly, in the plant, *Viridiplantae,* kingdom, only 3 complete genomes, 331 chromosomes, 625 scaffolds, and 394 contigs are annotated. The advances in next-generation sequencing technologies and the decrease in the cost of sequencing a complete transcriptome is driving a new era in which annotation will be increasing, important and productive.

A review of published manuscripts since 2012 [2-10] reveals that many developed pipelines have a common core component and use the NCBI BLAST tools [11] to align assembled transcriptomes against annotated databases of nucleotides or proteins to identify similarity and infer function. After an assembly, these alignments are the initial step to identify close and/or distant homologous genes, proteins, and functional domains that could be cross-referenced with other public databases, such as Gene Ontology [12], to generate new annotations of the query sequences. As the number of transcripts assembled per study increases, the computing power and storage required to align these transcripts to the BLAST databases also increases. On premise computer infrastructures (including server farms) have been used mainly for the computation of sequence alignments using BLAST. Many laboratories, however, are not equipped with the compute power required for the analysis of increased transcriptome sequencing results. Although a minimum infrastructure could be easy to build, it may be unnecessary with the advent of cloud computing and its utilization in computational biology.

Cloud computing offers an *on-demand* model where a user can dynamically allocate “unlimited” compute resources and then release them as soon as the analysis is complete [13]. There are many public cloud providers. Amazon Web Services (AWS; Seattle, WA, USA) and Google Cloud Platform (GCP; Mountain View, CA, USA) are popular examples. They offer a reduced cost of compute resources and a friendly user interface that makes them accessible for large computational biology experiments such as transcriptome annotation. In addition, private genomic cloud providers, for instance DNAnexus, Seven Bridges, and BT Cloud Compute, also are in the market and offer cloud-based genomics frameworks in line with the regulations that govern these field [14]. Although these commercial cloud providers make the execution of computational biology experiments easier, they also create additional charges for users. Some of these charges, however, can be discounted or avoided using public cloud providers.

Modern cloud providers offer "unlimited" compute resources that can be accessed *on-demand*. An *instance*, as the virtual machines are named in the cloud environment, is deployed using a variety of operating systems like GNU/Linux or Microsoft Windows. Users pay only for the time that the instance is running plus the cost of other resources such as network egress or the size of network storage devices. On a manually created instance, a workflow can be deployed but this is not cost efficient as the instance will need to be manually configured with the workflow dependencies. It will also remain active once the analysis is completed wasting resources. Conveniently, most cloud providers offer a batch system that can do the configuration automatically allowing users the submission of several parallel jobs. The batch system makes the process of instance creation, setup and termination fully automatic.

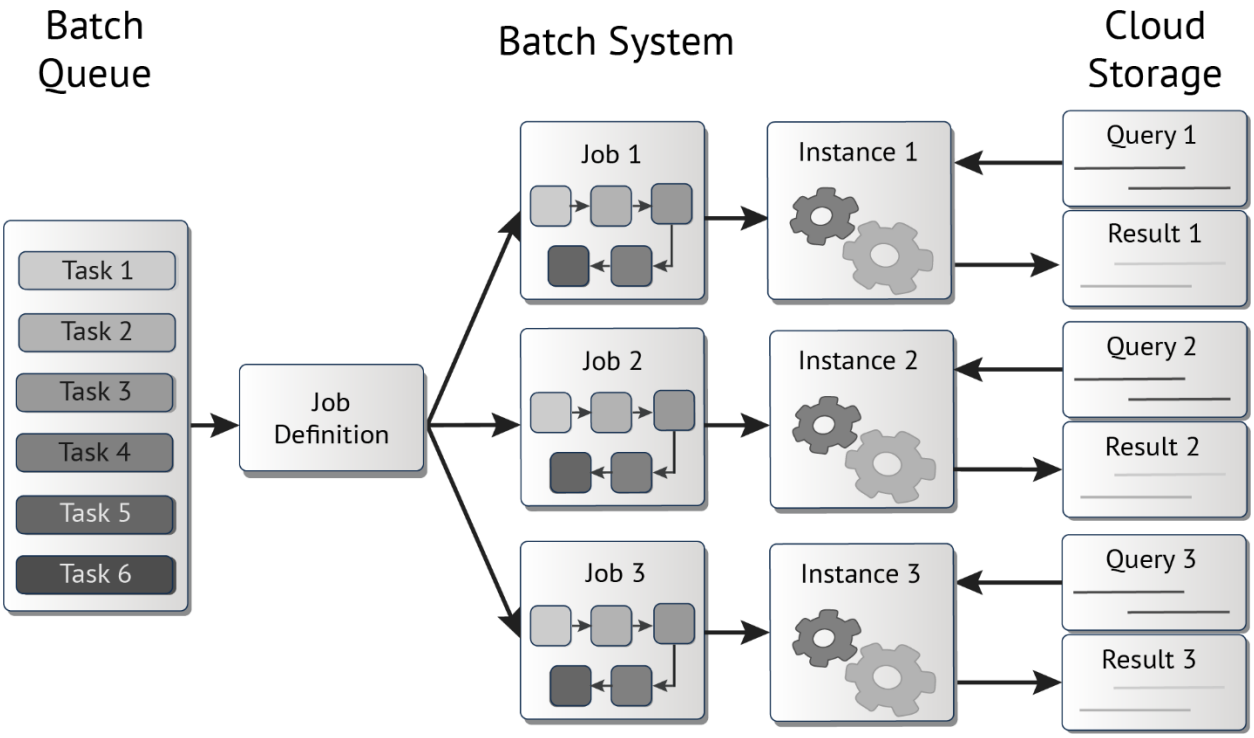


Figure 1: Basic components in a cloud-based batch system

Batch processing is a technique of processing data in one large group instead of individually. It reduces user interactivity to process submissions, making all remaining steps automatic. Modern cloud providers offer a batch system that can be personalized to process any kind of workflow. Figure 1 shows the component of a generic cloud batch system. It is comprised of a *batch queue* to which users submit the *tasks*. Each *task* uses a *job definition* to create a *job* where all computational resources and the workflow steps are outlined. Then, an *instance* is automatically created with the resources requested by the *job*. Since all the data for the analysis is in the cloud, the *instance* downloads the input data from the *cloud storage system* and, after successfully completing the workflow, uploads the results, releasing all computational resources.

The utilization of public cloud providers for computational biology experiments and analyses is increasing [15-18], however, little has been published describing cloud costs and implementation best practices. To address this issue, we present a comparative study of multiple BLAST alignments required in a transcriptome annotation project using two public cloud providers, AWS and GCP. This study aims to establish a general idea of the cost and time needed for the

execution of multiple BLAST searches. Our recommendation on best practices for deploying computational biology workflows in the cloud are presented.

Methods

Transcriptome Annotation Workflow

This study focuses on multiple BLAST alignments which are the most compute-demanding core of the transcriptome annotation process. BLAST alignments require considerable compute resources and they generate the intermediate results that are used to complete the annotation. The remaining part of the annotation pipeline was excluded from our study as it can be executed in a workstation and does not require an extensive use of the cloud.

The workflow uses as input a transcriptome in FASTA format. First, TransDecoder [19] is executed to generate all open reading frames (ORFs) from the input file. Then, BLASTP and RPS-BLAST are executed generating a list of homologous proteins and conserved domains. BLASTP uses the BLAST ***nr*** database, and RPS-BLAST uses the NCBI Conserved Domain Database (CDD) [20]. On the other side, BLASTN and RPST-BLASTN are executed using the BLAST ***nt*** database and the NCBI CDD database, respectively. These processes generate a list of homologous genes and a list of conserved domains, see Figure 2. The workflow was implemented using the Common Workflow Language (CWL) [21] and is freely available at: https://github.com/ncbi/cloud-transcriptome-annotation/blob/master/bin/cwl-ngs-workflows-cbb/workflows/Annotation/transcriptome_annotation.cwl

The workflow uses as input a FASTA file, which we named *query*, and includes multiple transcripts to be processed. The number of transcripts to be included in a *query* is another parameter that merits an analysis. The size of the *query* affects the whole workflow processing time as a complete transcriptome could be comprised of thousands to hundreds of thousands of transcripts assembled from a next-generation sequencing (NGS) experiment [22].

Our analysis is based on the execution of the workflow with a batch system provided by each cloud platform. This approach keeps the compute time and therefore the cost, to a minimum. It also limits the user interaction with the jobs to only the submission step.

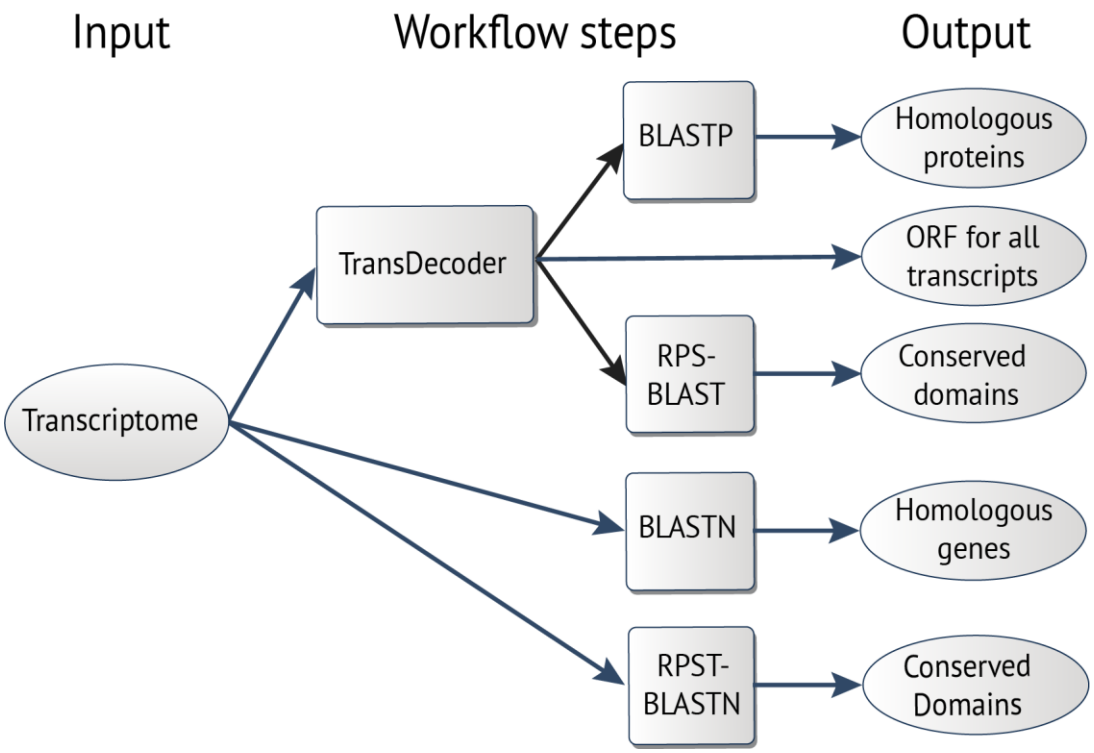


Figure 2: Schema of the transcriptome annotation workflow

Containerized workflows

Containerizing a workflow involves bundling it with all its dependencies and configuration files so that it can be executed across different computing environments. The workflow dependencies in the container uses the same version and compiled libraries when it is executed in any computing infrastructure which would make the process highly reproducible. In this study, we use Docker as the container engine. Docker permits the creation of container images that can be used on a personal laptop or on a cloud platform. The workflow container image generated is freely available from the Google Container registry (https://cloud.google.com/container-registry) with name: *gcr.io/cbb-research-dl/transannot-cloud-cmp*

All files used to generate this image are available at: https://github.com/ncbi/cloud-transcriptome-annotation/tree/master/config/gcp/docker

GCP

The Google Cloud Platform (GCP) offers a batch system specifically designed for life sciences, the Cloud Life Sciences (https://cloud.google.com/life-sciences). This system was initially Google Genomics but has evolved to allow the scientific community to process biomedical data at scale.

Cloud Life Sciences offers an Application Program Interface (API) implemented for users to develop their own workflow in JSON format using three main attributes: *actions*, *environments* and *resources*. *Actions* are the list of *commands* to execute using a defined container image. They also include statements to mount local solid-state drives (SSD) or network storage devices, defined in *resources*. *Environments* define the environment variables available inside the container. Finally, *resources* define the instance type and the local SSD or network storage devices.

Box 1: Brief extract of the GCP pipeline definition JSON file

```
{
  "actions": [
    ...,
    {
      "commands": [
        "/bin/bash",
        "-c",
        "cwltool --no-container --on-error continue --tmpdir-prefix /data/ --tmp-outdir-prefix /data/ --outdir /data/${SAMPLE}
https://raw.githubusercontent.com/ncbi/cloud-transcriptome-annotation/master/bin/cwl-ngs-workflows-
cbb/workflows/Annotation/transcriptome_annotation.cwl
 --blast_db_dir /data --threads ${CPUs} --evalue 1e-5 --blast_nt_db nt --blast_nr_db nr --blast_cdd_db split-cdd --fasta
/data/${SAMPLE}.fa >> /data/pipeline.log 2>&1"
      ],
      "imageUri": "gcr.io/cbb-research-dl/transannot-cloud-cmp",
      "mounts": [
        {
          "disk": "gcloud-shared",
          "path": "/data"
        }
      ]
    },
    ...,
    "environment": {
    "CPUs": "64"
  },
    "resources": {
    "virtualMachine": {
      "bootDiskSizeGb": 60,
      "bootImage": "projects/cos-cloud/global/images/family/cos-stable",
      "disks": [
        {
          "name": "gcloud-shared",
          "sizeGb": 600,
          "type":"local-ssd"
        }
      ],
      "machineType": "n1-standard-64",
      ...,
    }
  }
}
```

The API, using the JSON described in Box 1, automatically creates instances on-demand, following the requirements defined in the *resources* section of the job JSON file. GCP also provides a customized container image where the instance interacts with other GCP products like Google Storage where data is stored. In addition, GCP creates the instances using a customized Linux operating system that formats and mounts the instance local disks making them available for the jobs.

Box 1 also shows a brief extract of the pipeline used in GCP. We show only the main activity where the *command* attribute defines the command line to execute the CWL workflow. *ImageUri*

attribute defines the container image used to run the command. In this case, our previously created Docker image. Finally, the *mounts* attribute defines the paths in the container to mount the disks created in the *resources* attribute.

The *VirtualMachine* attribute defines the resources used to create the job instance. In this attribute, users can define instance boot disk size, operating system, extra disks and the machine type. The complete JSON file is available at: https://github.com/ncbi/cloud-transcriptome-annotation/blob/master/config/gcp/pipeline.json

AWS

AWS Batch (https://aws.amazon.com/batch/) is the batch system provided by Amazon. It is comprised of *compute environments*, *job queues* and *job definitions*. The *compute environment* defines the computational resources to be used by the batch jobs. It is connected to the Amazon Elastic Container Service (ECS) which is a fully managed service that creates and manages computer clusters inside the Amazon cloud environment. The resources defined by the compute environment are used by the ECS to create and setup instances in which the workload is distributed. *Job queues* are used as an intermediate service to associate submitted jobs with the compute environments. Lastly, the jobs use a *job definition*, in JSON format, which defines specific information for the job, like container images, commands, number of vCPUs, RAM memory, environment variables and local or remote folder to mount on the container.

Box 2 shows a brief extract of the job definition JSON script used in AWS. The *containerProperties* attribute defines the job properties. *Image* defines the container image, in this case our Docker image. *Command* defines the command to be executed inside the container. In the case of AWS, a single command can be outlined in the job definition, thus, complex pipelines with multiple steps can be encapsulated in a BASH script. This script can be stored inside the container image or the container can download it at runtime. For simplicity, we have included this script inside the Docker image.

The AWS Batch system automatically creates all infrastructure, network components and compute instances, following the requirements of the compute environments. The default configuration of the Amazon Machine Image (AMI) used for the instances, however, is not configured to use local SSD disks available on certain machine types. This limits the default options on the AWS Batch system to certain types of workflows. Workflows that use intensive disk IO operations will have improved performance and efficiency if local SSD disks are used. Thus, a modified AMI capable of use the instance local disks is required for our study. We create a customized AMI for our study that is freely available in the AWS zone *us-east1* with ID: *ami-0dac0383cac1dc96e*. This AMI creates an array with the local SSD disks in the instance using the Linux utility *mdadm*. The array is formatted with XFS filesystem and mounted in a folder named */data*.

Box 2: Brief extract of the AWS job definition JSON file

```
{
  ...,
  "containerProperties": {
    "image": "gcr.io/cbb-research-dl/transannot-cloud-cmp",
    "vcpus": 64,
    "memory": 131072,
    "command": [
      "/usr/envs/transannot/bin/aws-pipeline.sh"
    ],
    "volumes": [
      {
        "host": {
            "sourcePath": "/data"
        },
        "name": "data"
      }
    ],
    "environment": [
      {
        "name": "CPUs",
        "value": "32"
      }
    ],
    "mountPoints": [
      {
        "containerPath": "/data",
        "sourceVolume": "data"
      }
    ],
    ...,
  }
}
```

In addition, to improve the default AWS Batch options, Amazon offers a Virtual Private Cloud (VPC) that allows an extra layer of isolation for the resources used by the AWS Batch system. This VPC logically isolate all resources uses in a defined virtual network improving the security and it is customizable for the compute problem.

The templates used in our study to create all components of the AWS Batch system are available at: https://github.com/ncbi/cloud-transcriptome-annotation/tree/master/config/aws but all resources are created in the Jupyter notebook: "02 - AWS-Batch".

Jupyter Notebooks

Jupyter notebooks are an open-source web application framework that allows the creation and sharing of documents that contain live code [23]. It is a standard way to share scientific code for ease of reproducibility and reuse [24]. The implementation of our study was fully developed in Jupyter notebooks. Readers can reproduce our results and figures using the notebooks that are available at the project GitHub repository. The notebooks create all cloud resources and submit the jobs to the batch systems. They also retrieve the job logs in JSON format and create the figures automatically from those logs. Each notebook includes a description about its purpose and is named using a numeric prefix to highlight the execution order.

Results and Discussion

In this study, we present an analysis of the complexity, cost and best practices for executing the core components of a transcriptome annotation workflow in the cloud. We selected two popular cloud providers: GCP and AWS for our experiments. For each cloud provider, similar compute instances were tested using 16, 32 and 64 vCPUs. The machine types and their resources are described in Table 1.

Table 1: Machine types with resources in each cloud.

Provider	Machine type	vCPU	Memory (GB)	Instance Local SSD (GB)	Network Bandwidth (Gbps)	$/Hour
AWS	m5d	16	64	2 x 300	Up to 10	0.904
AWS	m5d	32	128	2 x 600	10	1.808
AWS	m5d	64	256	4 x 600	20	3.616
AWS	m5dn	16	64	2 x 300	Up to 25	1.088
AWS	m5dn	32	128	2 x 600	25	2.176
AWS	m5dn	64	256	4 x 600	75	4.352
GCP	n1	16	60	24 x 375	32	0.861
GCP	n1	32	120	24 x 375	32	1.393
GCP	n1	64	240	24 x 375	32	2.475
GCP	n2	16	64	24 x 375	32	0.951
GCP	n2	32	128	24 x 375	32	1.572
GCP	n2	64	256	24 x 375	32	2.816

We used the transcriptome assembled from a public BioProject with ID PRJNA320545 for the organism *Opuntia streptacantha*. The transcriptome includes 474,563 transcripts generated with Trinity [25], and is available in *data/PRJNA320545/transcriptome.fasta.gz*. From this pool of transcripts, we analyzed three types of *query* sizes, 2,000, 6,000, and 10,000 transcripts in each input *query* file. For each *query* size, we created 20 different FASTA files (input files for the workflow) (see notebook “01 - Data Partitioning”). Each of these files were submitted independently as jobs to the batch systems on each cloud provider.

Jobs were submitted to each cloud platform using the notebook “02 - Google Cloud Platform” and “02 - AWS-Batch”. We should highlight that the notebooks use the command line API provided by each cloud system. In the case of GCP, we used Cloud SDK (https://cloud.google.com/sdk). For AWS, we used the AWS Command Line Interface (https://aws.amazon.com/cli/). In each notebook, the 20 files created for each query size were copied to the respective cloud storage system, either Google Storage or S3 followed by job submissions submitted for each configuration of machine type/CPU.

Four times were collected from the jobs: the total running time, the time to transfer the BLAST databases to the instance local SSD disk, the time executing the CWL workflow and the time for creation, setup and release of the instance. Figure 3 shows the collected times for the 10,000

query size. Figure 3a shows the total running time for each input file (each containing 10,000 transcripts) for a total of 200,000 transcripts processed on each cloud provider, machine type and number of vCPUs.

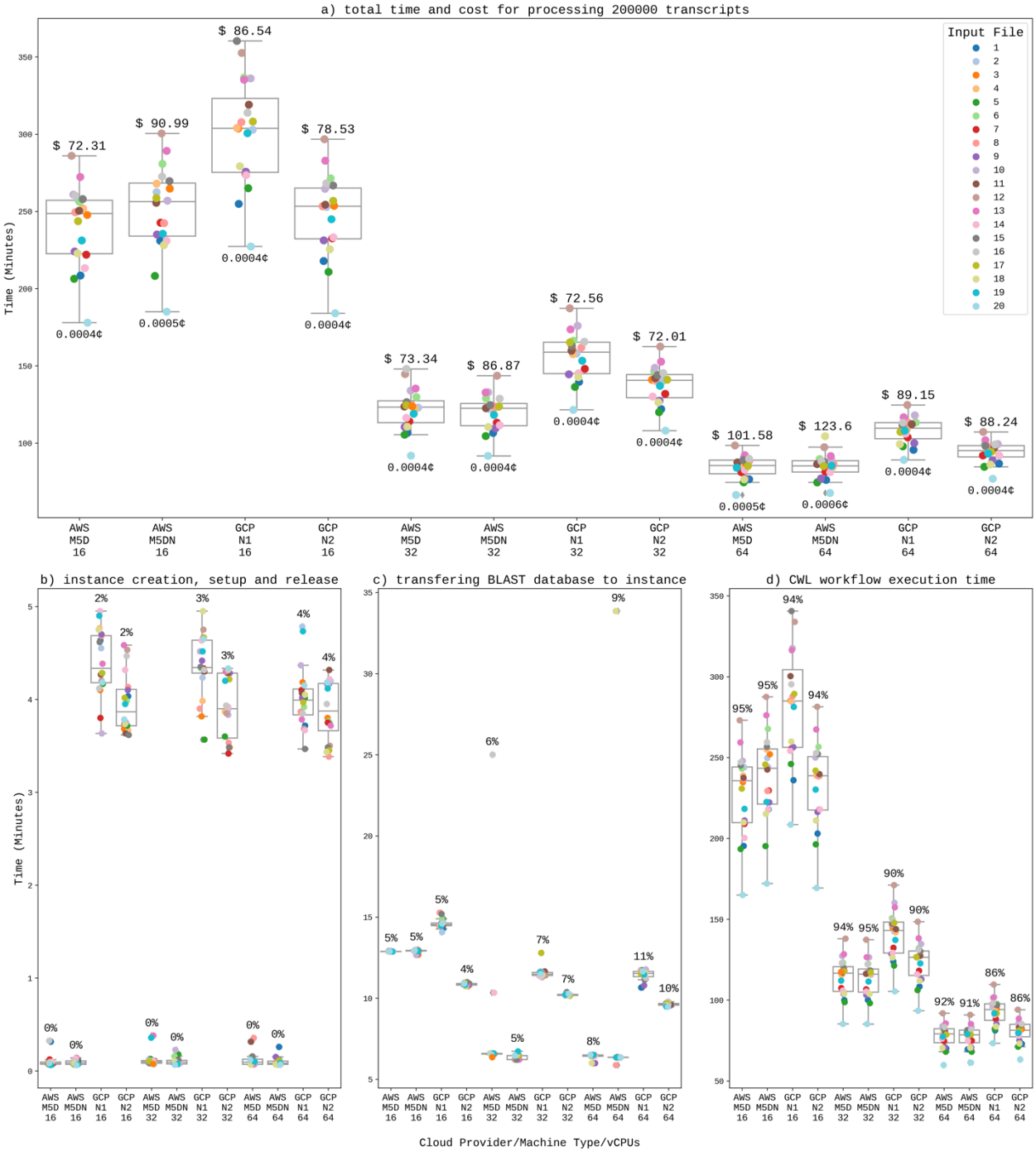


Figure 3: Breakdown of the time and cost for the 10,000 query size files. a) Total time for each input file for each configuration (Cloud provider/Machine Type/vCPUs). The total cost of processing the 20 input files (200,000 transcripts in total) is at the top of each box and the cost of processing one transcript is at the bottom of each box. b) Time and percent of the total cost for instance creation, setup and release. c) Time and percent of the cost for transferring the BLAST databases to the instance from the cloud storage bucket (S3 in AWS and Cloud Storage in GCP). d) Time and percent of the cost for the CWL workflow execution.

In addition, each box in Figure 3a shows the total cost for the 20 files (top) and the cost of processing one transcript (bottom). The bottom row with three plots shows the remaining three times collected from the jobs.

The total running time for the 10,000-query sized files are similar for the same number of vCPUs notwithstanding the cloud provider. Furthermore, this example shows how the running time can be reduced by more than a half by increasing the number of vCPUs. Unfortunately, this time reduction does not decrease the total cost of the project as the price per hour for machines with more vCPUs increases as well.

The AWS platform is more efficient than the GCP during the instance creation, setup and release, see Figure 3b. This stage takes only 0.1% of the total cost. The GCP cost for this stage goes from 1.5% to 4.5% on bigger machines. The differences are due to the AWS ECS which allocates new jobs on existing instances as soon as the instance gets free without releasing them, whereas GCP creates, sets up and releases an instance for each job.

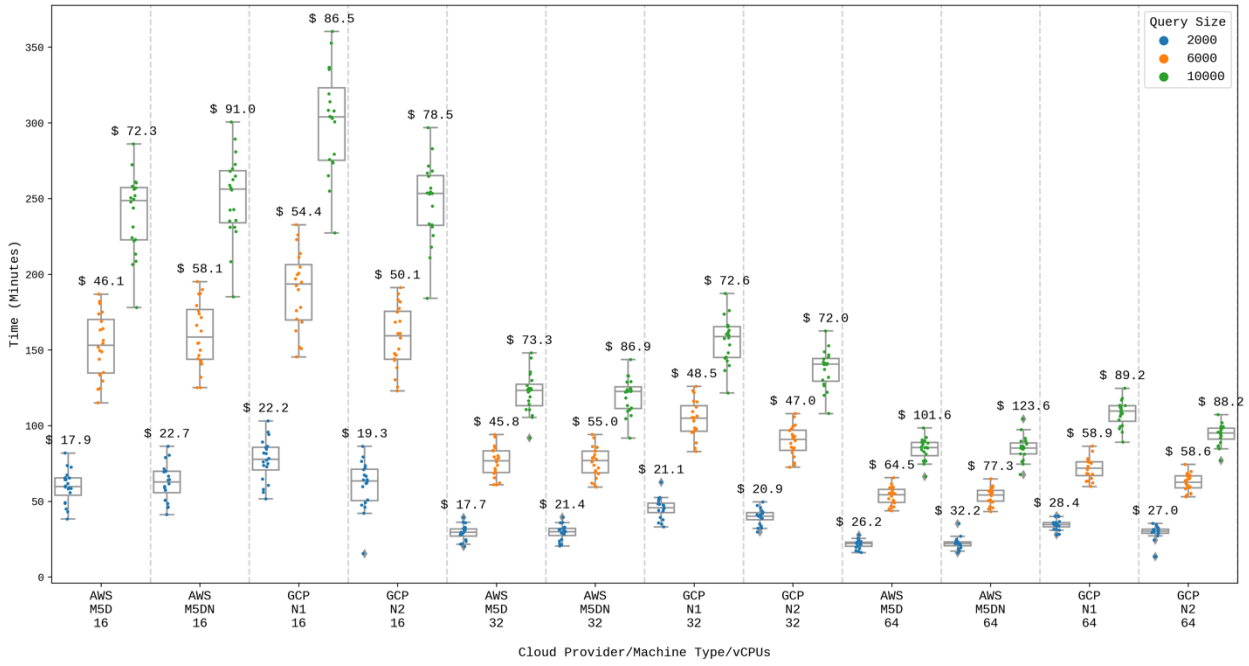


Figure 4: Total processing time for all configurations and query sizes

Transferring the BLAST databases from each cloud storage (S3 in AWS and Cloud Storage in GCP) bucket, Figure 3c, (current size is 342GB), to the instance local SSD disk is a crucial step in reducing the cost of the analysis. We tested the default parameters in both cloud providers which use network storage devices taking on average 1 hour which is about 30 % of the total cost of the analysis and which takes more time than the CWL workflow execution. After customizing both batch systems to use the instance local SSD disks, the time was reduced to a range of 4 % to 11 % of the total cost in the 10,000 query size.

As expected, the CWL workflow execution time is the most time-consuming part of the job, Figure 3d. All configurations show similar times for executing the CWL workflow. The GCP N1

machine type spent more time on the CWL workflow than the other machine types in all configurations because the GCP N1 is the Google first generation machine type with slower vCPUs than the other three machine types tested in this study.

Figure 4 shows all configurations tested with all query sizes. This plot shows that for smaller query sizes (2,000 in this case), there is no need to use machines with a large number of vCPUs as the difference in processing time is limited to a few minutes. For processing a large number of transcripts, it is best to use bigger query sizes and machines with more vCPUs as it reduces the cost of the complete annotation process.

Reducing the number of transcripts per input file will reduce the total running time but will also increase the cost of the project as more instances will be in used. For example, our experiment with the 10,000 query size processed 200,000 transcripts in 94 minutes with a total cost of $88.20 using 20 instances (GCP, N2, 64 CPU). Processing the same number of transcripts with a query size of 2,000 will cost $135.00 with all transcripts processed in 29 minutes using 100 instances (GCP, N2, 64 CPU) at the same time.

We have determined that a transcriptome with ~500,000 transcripts can be processed in less than 2 hours with a computing cost from US$200.00 to US$250.00. In our opinion, this is a reasonable cost and makes the transcriptome annotation process accessible to any genomic laboratory without access to an on-premise computational infrastructure.

Best practices

Our recommendation for best practices using public cloud providers for computational biology experiments are:

1. Write the pipeline using a workflow language. (We recommend CWL because the resulting product is portable and scalable, and it can be executed across a variety of computational environments as dissimilar as personal laptops or the cloud.)
2. Containerize the CWL workflow with Docker and use Conda/Bioconda to install all Bioinformatics tools in the container image.
3. Use Jupyter Notebooks for coding and documenting each step during experiments.
4. Use the cloud provider batch system for deploying jobs.
5. Execute a small test in the cloud to find the best instance type for a workflow.
6. Use the instance local disks for computing instead the default network devices.

Conclusion

Despite the differences in the configuration and setup of batch systems between GCP and AWS, the cost and processing time are similar for the type of workflow we designed. In our opinion, the choice of a cloud platform is not dependent on the workflow but, rather, on the specific details of the cloud provider. These specific details are related to the cloud platform's accessibility for institutional use, the user's technical knowledge of the specific platform service, or the availability of open-source frameworks to deploy the workflow on a specific cloud provider.

We found that GCP is easier to use as it only requires a JSON file for batch processing whereas AWS needs a complete setup of all components. GCP is more suitable for daily data analysis work in research laboratories. On the other hand, AWS, once properly configured, is more

efficient in terms of machine creation, setup and release. The ECS can reuse instances reducing the cost for large data analysis projects. AWS is more suitable for large data analysis groups to establish a set of queues and compute environments for multiple pipelines.

Availability of supporting source code and requirements

Project name: Cloud Transcriptome Annotation

Project home page: https://github.com/ncbi/cloud-transcriptome-annotation

Operating system(s): Linux and MacOS

Programming languages: Python, BASH

Other requirements: Conda/Bioconda, Jupyter Notebook

CWL workflow: https://github.com/ncbi/cloud-transcriptome-annotation/blob/master/bin/cwl-ngs-workflows-cbb/workflows/Annotation/transcriptome_annotation.cwl

Abbreviations

AMI: Amazon Machine Images (AMI)

API: Application Program Interface

AWS: Amazon Web Services

CDD: Conserved Domain Database

CWL: Common Workflow Language

ECS: Amazon Elastic Container Service

GCP: Google Cloud Platform

NCBI: National Center for Biotechnology Information

ORFs: Open Reading Frames

SSD: Solid State Disk

VPC: Virtual Private Cloud

Competing interests

The authors declare that they have no competing interests.

Authors' contributions

RVA, LMR and DL contributed to the design of the annotation workflow and manuscript preparation. RVA designed, implemented and executed all cloud environments, configurations and experiments. All authors read and approved all versions of the manuscript.

Funding

This work was supported by the Intramural Research Program of the National Library of Medicine, National Center for Biotechnology Information at the National Institutes of Health.

Acknowledgements

We would like to thank:

NCBI BLAST Group: Christiam Camacho, Vadim Zalunin, Greg Boratyn, Ryan Connor and Tom Madden for their support with BLAST.

NCBI Cloud and System Group: Al Graeff, Brian Koser, Andrew Arensburger, Brad Plecs, Ron Patterson and Dima Beloslyudtsev for their support with the cloud platforms.

References

1. Sayers, E.W., et al., *Database resources of the National Center for Biotechnology Information.* Nucleic Acids Res, 2020. **48**(D1): p. D9-D16.
2. Al-Qurainy, F., et al., *Comprehensive Stress-Based De Novo Transcriptome Assembly and Annotation of Guar (Cyamopsis tetragonoloba (L.) Taub.): An Important Industrial and Forage Crop.* Int J Genomics, 2019. **2019**: p. 7295859.
3. Chabikwa, T.G., et al., *De novo transcriptome assembly and annotation for gene discovery in avocado, macadamia and mango.* Sci Data, 2020. **7**(1): p. 9.
4. Ji, P., et al., *Characterization of common carp transcriptome: sequencing, de novo assembly, annotation and comparative genomics.* PLoS One, 2012. **7**(4): p. e35152.
5. Torre, S., et al., *RNA-seq analysis of Quercus pubescens Leaves: de novo transcriptome assembly, annotation and functional markers development.* PLoS One, 2014. **9**(11): p. e112487.
6. Carruthers, M., et al., *De novo transcriptome assembly, annotation and comparison of four ecological and evolutionary model salmonid fish species.* BMC Genomics, 2018. **19**(1): p. 32.
7. Haas, B.J., et al., *De novo transcript sequence reconstruction from RNA-seq using the Trinity platform for reference generation and analysis.* Nat Protoc, 2013. **8**(8): p. 1494-512.
8. Bryant, D.M., et al., *A Tissue-Mapped Axolotl De Novo Transcriptome Enables Identification of Limb Regeneration Factors.* Cell Rep, 2017. **18**(3): p. 762-776.
9. Vera Alvarez, R., et al., *Workflow and web application for annotating NCBI BioProject transcriptome data.* Database (Oxford), 2017. **2017**.
10. Gamez, R.M., et al., *Banana (Musa acuminata) transcriptome profiling in response to rhizobacteria: Bacillus amyloliquefaciens Bs006 and Pseudomonas fluorescens Ps006.* BMC Genomics, 2019. **20**(1): p. 378.
11. Altschul, S.F., et al., *Basic local alignment search tool.* J Mol Biol, 1990. **215**(3): p. 403-10.
12. Ashburner, M., et al., *Gene ontology: tool for the unification of biology. The Gene Ontology Consortium.* Nat Genet, 2000. **25**(1): p. 25-9.
13. Langmead, B. and A. Nellore, *Cloud computing for genomic data analysis and collaboration.* Nature Reviews Genetics, 2018. **19**(4): p. 208-219.
14. Marx, V., *Genomics in the clouds.* Nature Methods, 2013. **10**(10): p. 941-945.
15. Peters, K., et al., *PhenoMeNal: processing and analysis of metabolomics data in the cloud.* Gigascience, 2019. **8**(2).

16. Belyeu, J.R., et al., *SV-plaudit: A cloud-based framework for manually curating thousands of structural variants.* Gigascience, 2018. **7**(7).
17. Kiar, G., et al., *Science in the cloud (SIC): A use case in MRI connectomics.* Gigascience, 2017. **6**(5): p. 1-10.
18. Hiltemann, S., et al., *CGtag: complete genomics toolkit and annotation in a cloud-based Galaxy.* Gigascience, 2014. **3**(1): p. 1.
19. Haas, B. and A. Papanicolaou. *TransDecoder (Find Coding Regions Within Transcripts)*. 2020; Available from: https://github.com/TransDecoder/TransDecoder/wiki.
20. Yang, M., et al., *NCBI's Conserved Domain Database and Tools for Protein Domain Analysis.* Curr Protoc Bioinformatics, 2020. **69**(1): p. e90.
21. Peter, A., et al., *Common Workflow Language, v1.0*. 2016.
22. Pertea, M., *The human transcriptome: an unfinished story.* Genes (Basel), 2012. **3**(3): p. 344-60.
23. Shen, H., *Interactive notebooks: Sharing the code.* Nature, 2014. **515**(7525): p. 151-2.
24. Perkel, J.M., *Why Jupyter is data scientists' computational notebook of choice.* Nature, 2018. **563**(7729): p. 145-146.
25. Grabherr, M.G., et al., *Full-length transcriptome assembly from RNA-Seq data without a reference genome.* Nat Biotechnol, 2011. **29**(7): p. 644-52.

Figure 1

Click here to access/download;Figure;Figure 1.png

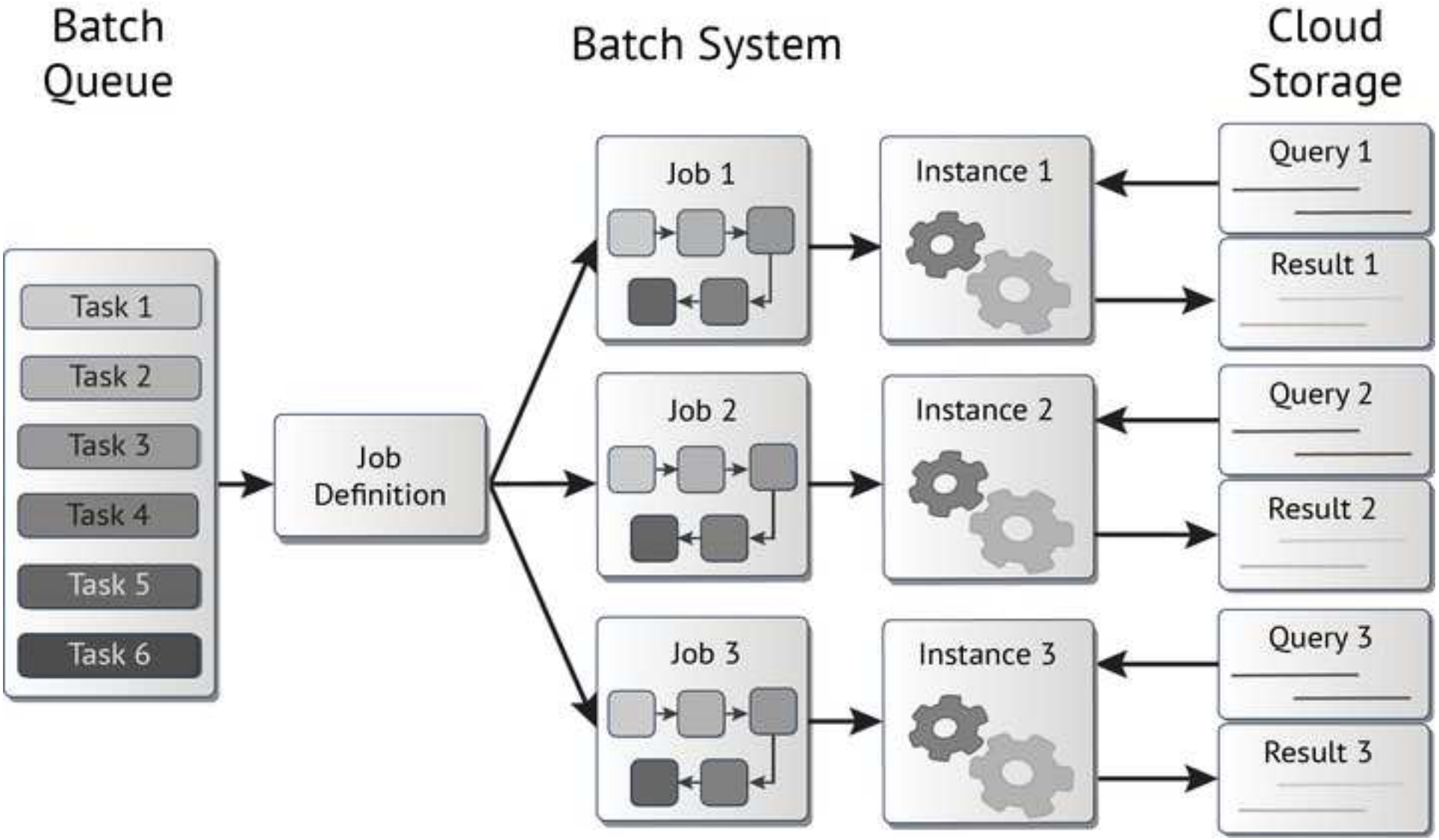


Figure 2

Click here to access/download;Figure;Figure 2.png

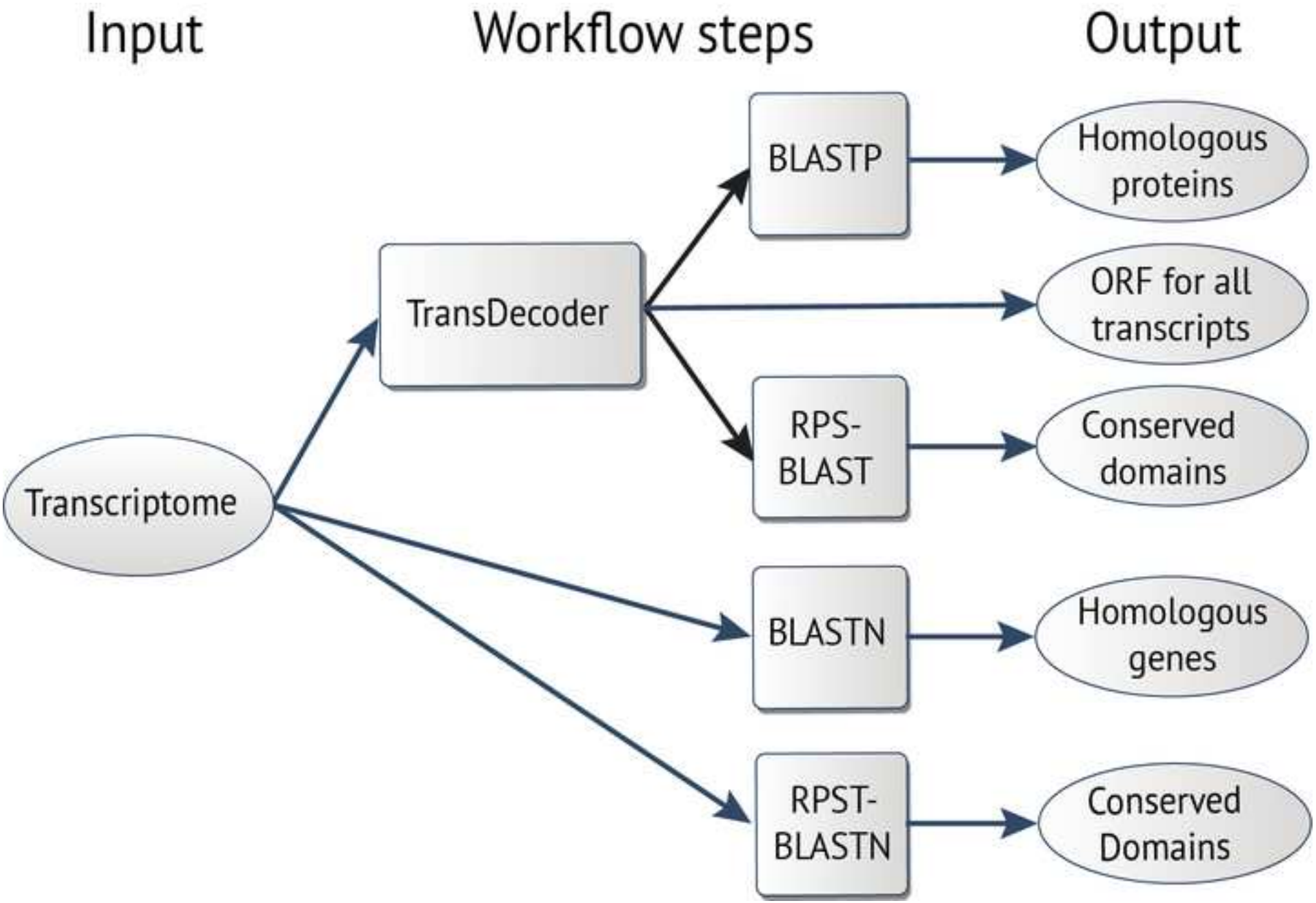


Figure 3

Click here to access/download;Figure;Figure 3.png

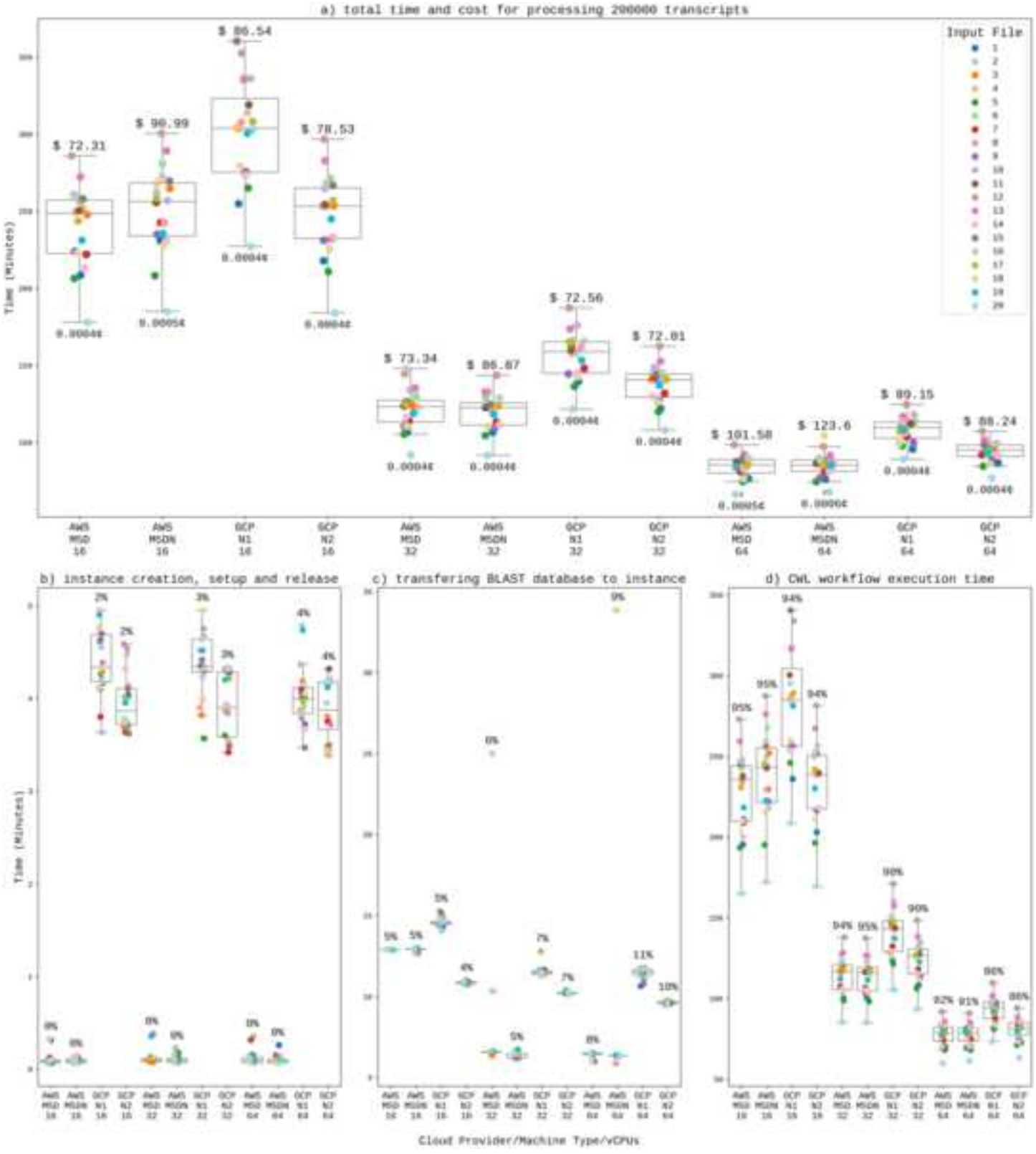
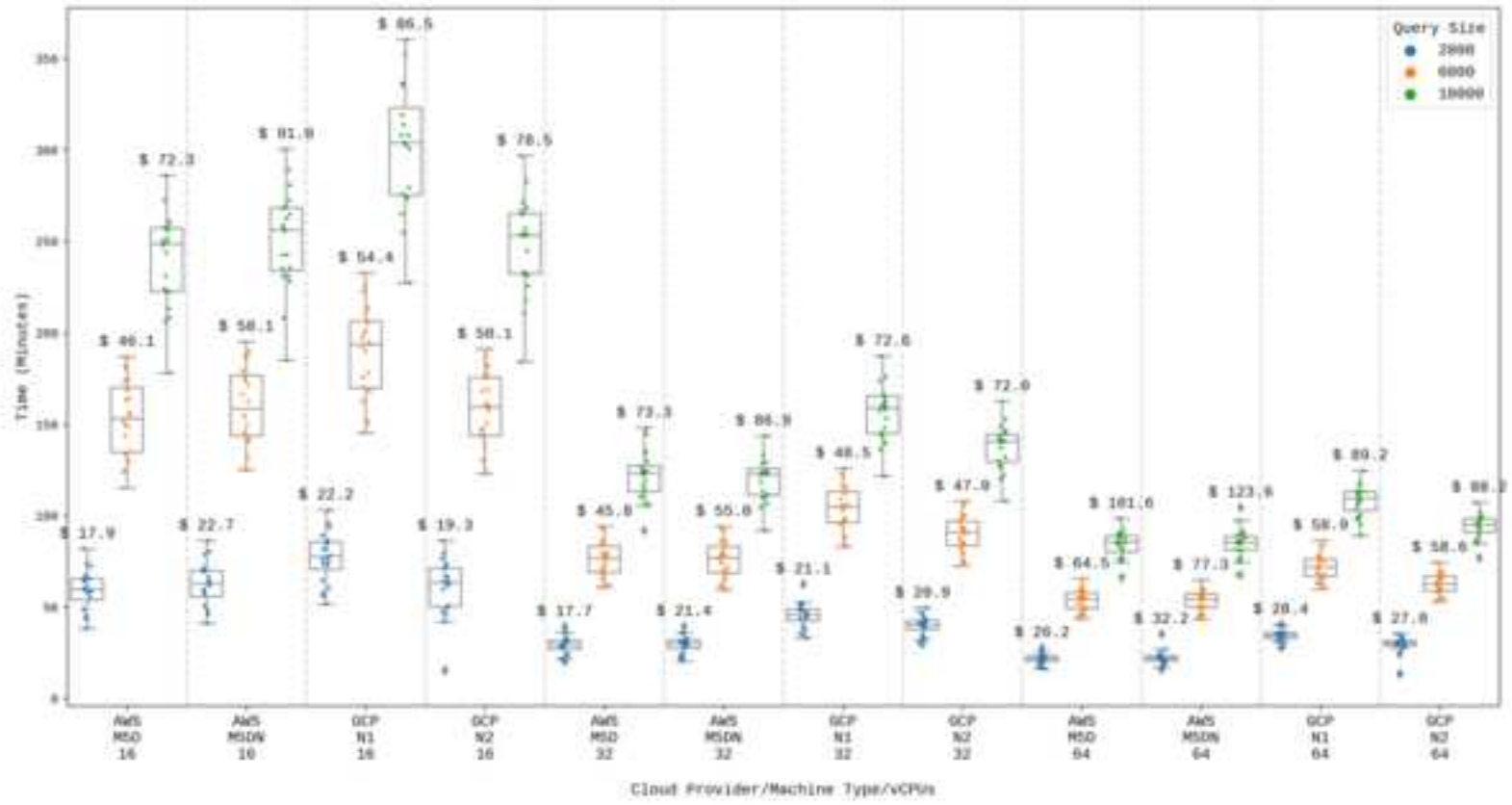


Figure 4

Click here to access/download;Figure;Figure 4.png



Personal Cover Click here to access/download;Personal Cover;Cover_letter.docx

DEPARTMENT OF HEALTH & HUMAN SERVICES Public Health Service

National Institutes of Health
National Library of Medicine
Bethesda, Maryland 20894

July 2, 2020

Dear Editors,

We would like to submit our manuscript entitled "Transcriptome annotation in the cloud: complexity, best practices and cost" for your consideration for publication in *GigaScience as a Technical Note*.

Transcriptome annotation with functional and biological processes is a complex analytical procedure that requires the integration of multiple databases and several advanced computational tools. This annotation is an important step in developing an understanding of the biological complexity of an organism. The advances in next-generation sequencing technologies and the decrease in the cost of sequencing a complete transcriptome is driving a new era in which annotation will be increasing, important and productive.

A review of published manuscripts since 2012 [1-9] reveals that many developed pipelines have a common core component and use the NCBI BLAST tools [10] to align assembled transcriptomes against annotated databases of nucleotides or proteins to identify similarity and infer function. However, BLAST alignments are computing workloads that demands computer power only available in High Performance Computing (HPC) infrastructures. Many laboratories, however, are not equipped with the compute infrastructure required for the analysis of increased transcriptome sequencing results. Although a minimum infrastructure could be easy to build, it may be unnecessary with the advent of cloud computing and its utilization in computational biology.

In this manuscript, we present a comparative study of multiple BLAST alignments using two public cloud providers: Amazon Web Services (AWS; Seattle, WA, USA) and Google Cloud Platform (GCP; Mountain View, CA, USA). We have prepared several Jupyter Notebooks with all the code required to submit BLAST jobs to the batch system on each cloud provider in order to reproduce or extend our results. We demonstrate that the public cloud providers are a practical alternative for executing advanced computational biology experiments at quite low cost. Using our cloud recipes, the BLAST alignments required to annotate a transcriptome with ~500,000 transcripts can be processed in less than 2 hours with a computing cost of about US$ 200-250.

All code used during this study is freely available at: https://github.com/ncbi/cloud-transcriptome-annotation

Thank you very much for your consideration of our manuscript. We look forward to your reply on this submission.

Sincerely,

Roberto Vera Alvarez [1,a], Leonardo Mariño-Ramírez [1,2,b] and David Landsman [1,c]

[1] Computational Biology Branch, National Center for Biotechnology Information, National Library of Medicine, Natioanl Insitutes of Health, Bethesda, MD, USA.
[2] Current address: Division of Intramural Research, National Institute on Minority Health and Health Disparities, NIH, Bethesda, MD, USA.

a. veraalva@ncbi.nlm.nih.gov
b. marino@nih.gov
c. landsman@ncbi.nlm.nih.gov

References

1. Al-Qurainy F, Alshameri A, Gaafar AR, Khan S, Nadeem M, Alameri AA, et al. Comprehensive Stress-Based De Novo Transcriptome Assembly and Annotation of Guar (Cyamopsis tetragonoloba (L.) Taub.): An Important Industrial and Forage Crop. Int J Genomics. 2019;2019:7295859. doi:10.1155/2019/7295859.

2. Chabikwa TG, Barbier FF, Tanurdzic M and Beveridge CA. De novo transcriptome assembly and annotation for gene discovery in avocado, macadamia and mango. Sci Data. 2020;7 1:9. doi:10.1038/s41597-019-0350-9.

3. Ji P, Liu G, Xu J, Wang X, Li J, Zhao Z, et al. Characterization of common carp transcriptome: sequencing, de novo assembly, annotation and comparative genomics. PLoS One. 2012;7 4:e35152. doi:10.1371/journal.pone.0035152.

4. Torre S, Tattini M, Brunetti C, Fineschi S, Fini A, Ferrini F, et al. RNA-seq analysis of Quercus pubescens Leaves: de novo transcriptome assembly, annotation and functional markers development. PLoS One. 2014;9 11:e112487. doi:10.1371/journal.pone.0112487.

5. Carruthers M, Yurchenko AA, Augley JJ, Adams CE, Herzyk P and Elmer KR. De novo transcriptome assembly, annotation and comparison of four ecological and evolutionary model salmonid fish species. BMC Genomics. 2018;19 1:32. doi:10.1186/s12864-017-4379-x.

6. Haas BJ, Papanicolaou A, Yassour M, Grabherr M, Blood PD, Bowden J, et al. De novo transcript sequence reconstruction from RNA-seq using the Trinity platform for reference generation and analysis. Nat Protoc. 2013;8 8:1494-512. doi:10.1038/nprot.2013.084.

7. Bryant DM, Johnson K, DiTommaso T, Tickle T, Couger MB, Payzin-Dogru D, et al. A Tissue-Mapped Axolotl De Novo Transcriptome Enables Identification of Limb Regeneration Factors. Cell Rep. 2017;18 3:762-76. doi:10.1016/j.celrep.2016.12.063.

8. Vera Alvarez R, Medeiros Vidal N, Garzon-Martinez GA, Barrero LS, Landsman D and Marino-Ramirez L. Workflow and web application for annotating NCBI BioProject transcriptome data. Database (Oxford). 2017;2017 doi:10.1093/database/bax008.

9. Gamez RM, Rodriguez F, Vidal NM, Ramirez S, Vera Alvarez R, Landsman D, et al. Banana (Musa acuminata) transcriptome profiling in response to rhizobacteria: Bacillus amyloliquefaciens Bs006 and Pseudomonas fluorescens Ps006. BMC Genomics. 2019;20 1:378. doi:10.1186/s12864-019-5763-5.

10. Altschul SF, Gish W, Miller W, Myers EW and Lipman DJ. Basic local alignment search tool. J Mol Biol. 1990;215 3:403-10. doi:10.1016/S0022-2836(05)80360-2.

NIH Publishing Agreement and Manuscript Cover Sheet
Click here to access/download;Personal Cover;NIH Publishing Agreement and Manuscript Cover Sheet.pdf

NIH Publishing Agreement & Manuscript *Cover Sheet*

By signing this Cover Sheet, the Author, on behalf of NIH, agrees to the provisions set out below, which **modify and supersede,** solely with respect to NIH, any conflicting provisions that are in the Publisher's standard copyright agreement (the "Publisher's Agreement"). If a Publisher's Agreement is attached, execution of this Cover Sheet constitutes an execution of the Publisher's Agreement, subject to the provisions and conditions of this Cover Sheet.

1. **Indemnification.** No Indemnification or "hold harmless" obligation is provided by either party.

2. **Governing Law.** This agreement will be governed by the law of the court in which a claim is brought.

3. **Copyright.** Author's contribution to the Work was done as part of the Author's official duties as a NIH employee and is a Work of the United States Government. Therefore, copyright may not be established in the United States. 17 U.S.C. § 105. If Publisher intends to disseminate the Work outside of the U.S., Publisher may secure copyright to the extent authorized under the domestic laws of the relevant country, subject to a paid-up, nonexclusive, irrevocable worldwide license to the United States in such copyrighted work to reproduce, prepare derivative works, distribute copies to the public and perform publicly and display publicly the work, and to permit others to do so.

4. **No Compensation.** No royalty income or other compensation may be accepted for work done as part of official duties. The author may accept for the agency a limited number of reprints or copies of the publication.

5. **NIH Representations.** NIH represents to the Publisher that the Author is the sole author of the Author's contribution to the Work and that NIH is the owner of the rights that are the subject of this agreement; that the Work is an original work and has not previously been published in any form anywhere in the world; that to the best of NIH's knowledge the Work is not a violation of any existing copyright, moral right, database right, or of any right of privacy or other intellectual property, personal, proprietary or statutory right; that where the Author is responsible for obtaining permissions or assisting the Publishers in obtaining permissions for the use of third party material, all relevant permissions and information have been secured; and that the Work contains nothing misleading, obscene, libelous or defamatory or otherwise unlawful. NIH agrees to reasonable instructions or requirements regarding submission procedures or author communications, and reasonable ethics or conflict of interest disclosure requirements unless they conflict with the provisions of this Cover Sheet.

6. **Disclaimer.** NIH and the Author expressly disclaim any obligation in Publisher's Agreement that is not consistent with the Author's official duties or the NIH mission, described at http://www.nih.gov/about/. NIH and the Author do not disclaim obligations to comply with a Publisher's conflict of interest policy so long as, and to the extent that, such policy is consistent with NIH's own conflict of interest policies.

7. **For Peer-Reviewed Papers to be Submitted to PubMed Central.** The Author is a US government employee who must comply with the NIH Public Access Policy, and the Author or NIH will deposit, or have deposited, in NIH's PubMed Central archive, an electronic version of the final, peer-reviewed manuscript upon acceptance for publication, to be made publicly available no later than 12 months after the official date of publication. The Author and NIH agree (notwithstanding Paragraph 3 above) to follow the manuscript deposition procedures (including the relevant embargo period, if any) of the publisher so long as they are consistent with the NIH Public Access Policy.

8. **Modifications.** PubMed Central may tag or modify the work consistent with its customary practices and with the meaning and integrity of the underlying work.

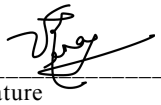
The NIH Deputy Director for Intramural Research, Michael Gottesman, M.D., approves this publishing agreement and maintains a single, signed copy of this text for all works published by NIH employees, and contractors and trainees who are working at the NIH. No additional signature from Dr. Gottesman is needed.

Author's name: Roberto Vera Alvarez, Leonardo Mariño-Ramírez, and David Landsman

Author's Institute or Center: NCBI/NLM/NIH Check if Publisher' Agreement is attached ☐

Name of manuscript/work: Transcriptome annotation in the cloud: complexity, best practices and cost

Name of publication: GigaScience



Author's signature

07/02/2020

Date