2016 IEEE International Parallel and Distributed Processing Symposium Workshops

A Scalable Pipeline For Transcriptome Profiling Tasks With On-demand Computing Clouds

Shayan Shams
Center for Computation and Technology
Louisiana State University
Baton Rouge, LA 70803
Email: sshams2@cct.lsu.edu

Nayong Kim
Center for Computation and Technology
Louisiana State University
Baton Rouge, LA 70803
Email: nykim@cct.lsu.edu

Xiandong Meng
Genomics Devision
Lawrence Berkeley National Laboratory
Department of Energy
Joint Genomic Institute
Walnut Creek, CA 94598
Email: xiandongmeng@lbl.gov

Ming Tai Ha
Electrical and Computer Engineering Busch Campus
Rutgers University
Piscataway, NJ 08854
Email: ming.tai.ha@gmail.com

Shantenu Jha
Electrical and Computer Engineering Busch Campus
Rutgers University
Piscataway, NJ 08854-8058
Email: shantenu.jha@rutgers.edu

Zhong Wang
Genomics Devision
Lawrence Berkeley National Laboratory
Department of Energy
Joint Genomic Institute
Walnut Creek, CA 94598
Email: zhongwang@lbl.gov

Joohyun Kim*
Center for Computation and Technology
Louisiana State University
Baton Rouge, LA 70803
Email: jhkim@cct.lsu.edu

***Abstract*—We introduce a pilot-based approach with which scalable data analytics essential for a large RNA-seq data set are efficiently carried out. Major development mechanisms, designed in order to achieve the required scalability, in particular, targeting cloud environments with on-demand computing, are presented. With an example of Amazon EC2, by harnessing distributed and parallel computing implementations, our pipeline is able to allocate optimally computing resources to tasks of a target workflow in an efficient manner. Consequently, decreasing time-to-completion (TTC) or cost, avoiding failures due to a limited resource of a single node, and enabling scalable data analysis with multiple options can be achieved. Our developed pipeline benefits from the underlying pilot system, Radical Pilot, being readily amenable to scalable solutions over distributed heterogeneous computing resources and suitable for advanced workflows of dynamically adaptive executions. In order to provide insights on such features, benchmark experiments, using two real data sets, were carried out. The benchmark experiments focus on the most computationally expensive transcript assembly step. Evaluation and comparison of transcript assembly accuracy using a single de novo assembler or the combination of multiple assemblers are also presented, underscoring its potential as a platform to support multi-assembler multi-parameter methods or ensemble methods which are statistically attractive and easily feasible with our scalable pipeline. The developed pipeline, as manifested by results presented in this work, is built upon effective strategies that address major challenging issues and viable solutions toward an integrative and scalable method for large-scale RNA-seq data analysis, particularly maximizing merits of Infrastructure as a Service (IaaS) clouds.**

I. Introduction

RNA-seq is one of the most widely adopted methods employing the high-throughput DNA sequencing technology (aka Next-Generation Sequencing and NGS in short)[1]. In spite of its remarkable successes in various applications for virtually all areas of life sciences, outstanding challenges still remain as roadblocks. Data analytics of this revolutionary approach are complicated. For example, the task of an accurate transcript assembly is non-trivial, significantly limiting its usages for non-model organisms[2, 3]. Also, many challenges are largely associated with technical aspects with NGS such as the short read length and various artifacts arising from sequencing errors, unknown sample heterogeneity, and unknown variations in data sets[4]. Interestingly, the rapid increase of the sequencing data volume propelled by the falling sequencing cost as well as the widespread utilization, along with those challenges in data analytics, have been increasingly garnering intensive interests on NGS data analytics as one of Big Data problems. Indeed, the main motivation of this study is to develop effective strategies for a viable solution in the transcriptome profiling

* Corresponding author

task challenged by volume, heterogeneity, and complexity of both sequencing data and intermediate data generated in the middle of an analysis pipeline.

RNA-seq data analysis requires multiple different computational tasks, favoring increasingly an integrative approach composed of multiple underlying pipelines[5, 6]. Our main goal is to develop a new pipeline based on the existing tool, Rnnotator[5]. Rnnotator was originally designed as a perl-based tool for RNA-seq. The basic workflow of Rnnotator is to run multiple steps, corresponding to pre-processing of sequencing reads, transcript assembly, transcript quantification and differential gene expression (only optional for cases when multiple sample conditions are provided)(see Fig. 1). While the majority of tools in RNA-seq employ a reference-based method, Rnnotator, developed even in relatively early days in this field, is one of few tools that put the transcript assembly as a key strategy. The transcript assembly step is implemented with the multiple k-mer approach for which each k-mer calculation is conducted with an existing de novo assembler among Velvet, Oases, IDBA, Minia, and Ray as of this writing. Whereas many projects including those from Joint Genomic Institute (JGI) have been utilizing the tool successfully[2], the original architecture is primarily intended with the use of large-scale HPC systems such as those available from the National Energy Research Scientific Computing Center (NERSC). Consequently, the usability of the original Rnnotator is, so far somehow, limited with the requirement of the use of specific types of HPC systems in which the local scheduler like Sun Grid Engine (SGE) plays a key role for parallelization. Its perl-based design has also limitations in scalability for growing data sets and for its performance due to the lack of supporting distributed tasks and flexible execution scenarios.

To address such limitations, our new approach changed the core architecture of Rnnotator with the pilot framework, Radical Pilot. The pilot framework is developed by Jha and his coworkers, with which an efficient management of sub tasks constituting an entire workflow is readily achieved over distributed computing systems[7]. Previously, we demonstrated various workflow scenarios driven by the pilot framework for a variety of scientific applications including sequencing data analytics[8–10].

Facilitated by RP, the developed pipeline adds novel features which are not easily feasible with non-scalable conventional approaches. Specifically, we aim to improve the transcript assembly task which is the major computationally demanding one in RNA-seq pipelines including the original Rnnotator tool. This task is not only complicated algorithmically, but also difficult to be scalable. Since we focused on the solution for large sequencing data, we exploit the current Rnnotator strategy, the use of scalable de novo assemblers that runs on distributed resources. Note that among all assemblers available, Ray is the only assembler that supports scalable tasks with its Message Passing Interface (MPI) support. First, we added the support of the two other assemblers, ABySS, another MPI-based assembler, and Contrail, a Hadoop MapReduce-based assembler. We conducted benchmark experiments for the performance of the transcript assembly with these assemblers. The benchmark results highlight underlying factors for optimal solutions of the assembly step with the multi-node capable tools based on MPI and Hadoop.

The need of a scalable solution is also associated with the complexity of NGS data analytics. Considering artifacts and errors, a single tool is unlikely to manage all unknown factors buried or uncertain among observed or processed data sets. Indeed, there exists strong consensus to defy a common practice relying upon a single tool across numerous comparative studies, including those for sequence assembly[11–13]. However, due to the requirement of complex computation and costs, the better option that utilizes multiple analyses with multiple tools or different parameters is not practically favored. Interestingly, combining all results from such multiple tools and parameters with a statistically sounding method seems to be attractive. For example, recent studies exploiting the ensemble or the consensus method for assembly suggested better outcomes[13–15]. In fact, a large number of studies in many areas including statistical learning manifested that ensemble methods often outperformed methods based on the single model or parameter[16].

Overall, motivated by the need of scalable solutions, a new type of a pipeline has been developed for RNA-seq and we present its details below.

The paper is organized as follows. In the following section, related works are summarized underscoring key common interests between them and our work. Then, we describe backgrounds of our pipeline introducing the Rnnotator workflow, Radical Pilot, and EC2, the main focused transcript assembly, and data sets used. The section for results and discussion follows. The paper ends with concluding remarks containing future plans.

II. Related works

Among a plethora of tools relevant to our work, we focus on tools specifically related with the two aspects. The first aspect is whether a tool provides an integrative approach for RNA-seq data analysis, specifically with cloud computing. The second aspect is whether a tool employs a software framework for distributed computing. It is not an easy task to survey all related tools, and thus our list is somehow intended to provide a general overview on related works.

A growing evidence for increasing interest on cloud-based applications for RNA-seq has been seen as described in the recent article[6]. Myrna is a tool to use cloud computing for RNA-seq analysis[17]. FX is a tool for RNA-seq that employs Hadoop with which the estimation of gene expression levels is possible[18]. RAP is a pipeline for RNA-seq using cloud computing and web techniques[19]. Stormbow was introduced for large scale expression quantification on Amazon cloud[20].

Compared to an effort to develop a standalone tool, utilizing a distributed computing framework has many benefits as summarized in the article[21] For example, a framework for distributed genome assembly was introduced and demonstrated

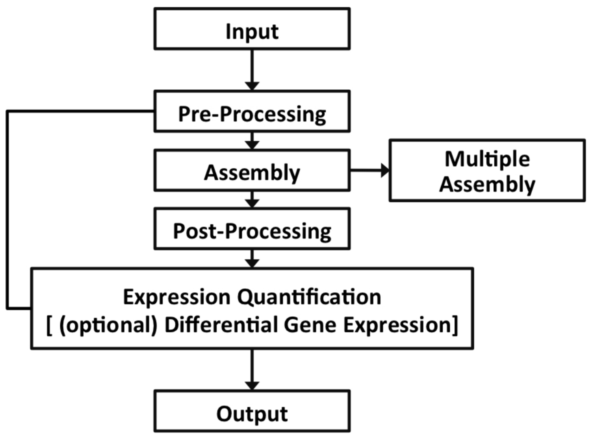

Input
Pre-Processing
Assembly
Multiple Assembly
Post-Processing
Expression Quantification
[(optional) Differential Gene Expression]
Output

Fig. 1. The workflow of the Rnnotator pipeline is schematically shown.

the speed up with various computer resources from cluster, clouds, and grids[22]. GATK[23] is a widely known tool for which the framework based on cloud-friendly MapReduce constitutes its core part. In fact, MapReduce and related programming models have been utilized for achieving scalable goals with clouds. Examples include a Hadoop-based genome assembler, Contrail[24] and other numerous applications including SparkSeq[25] built upon SPARK[26].

Finally, iMetAMOS[13] is not a cloud-based tool and is not scalable over multiple nodes, but is similar to our pipeline, providing an integrated platform to support multiple assembly programs and validation tools together.

III. Background and methods

A. Rnnotator

Rnnotator[5] is a software suite mostly written in Perl, and its main workflow is basically composed of the four steps as shown in Fig. 1.

The primary feature is the transcript assembly that utilizes the multiple k-mer strategy, and an existing De Bruijn Graph (DBG) assembler is used for each k-mer assembly. Assembled contigs from different k-mer assemblies are then processed for identifying overlaps and merged with VMATCH[27] and Minimus2[28]. Currently, assemblers such as Velvet[29], Oases[30], Ray[31], IDBA[32], and Minia[33] can be used. Among them, Ray is the only assembler that can process a large size sequencing data owing to its implementation with MPI.

The original structure of Rnnotator is principally limited for scalable analysis to deal with large sequencing data. Distributed computation is only supported in a limited manner. Again, de novo assemblers except Ray are not scalable, implying that the entire pipeline fails if a computing node running an assembly task has no sufficient memory footprint for dealing with the data set. The parallelization is limited in multi-threading in a single node or by the local scheduler, SGE if available. Not surprisingly, Rnnotator is not cloud-friendly, either. Likely, it has no option to effectively run on a heterogeneous cluster architecture or cloud environments with on-demand computing in which different types of instances or Virtual Machine (VM) can be utilized.

However, the overall architecture of Rnnotator is modular and well structured for possible extensions. Our long-term project plan is to develop Rnnotator-based service for transcriptome analysis that can be offered via the community-wide mechanism such as science gateways, with multiple flexible choice options and improved performance and scalability[34]. In this work, more specifically, we focus the support of large scale transcript assembly in the main pipeline allowing multiple assemblers to run on distributed resources. This new feature is systematically achieved by changing its underlying architecture from the Perl-based pipeline tool to the pilot-based pipeline architecture as described in details below.

B. Amazon EC2

Amazon EC2 is a commercial cloud and categorized as a Infrastructure-as-a-service (IaaS) cloud. On-demand computing provided by IaaS clouds such as EC2 or OpenStack has various merits as a computing environment for bioinformatics. For example, its virtualization mechanism permits to reuse an existing machine image in which all required tools and environmental variables are set up, resulting in efficient tool management and development. A cluster with multiple nodes is also easily built and thus can take advantages of existing experiences with High-performance Computing (HPC) clusters. EC2 also provides multiple instance types for individual nodes varying their computing power, memory, storage, accelerators such as GPU and networking performance, leveraging benefits from a heterogeneous system.

For the entire experiments reported in this work, we used two instance types, r3.2xlarge and c3.2xlarge. Both have 8 cores, and the cost of r3.2xlarge is 0.7 USD per hour and c3.2xlarge is 0.42 USD per hour. The memory of the former is 61 GB, while the latter is equipped with 16 GB.

C. Pilot-based workflow patterns

A distributed application is a class of scientific applications intrinsically composed of multiple sub-tasks whose execution is conducted over distributed computing resources and storage system. A pilot-based framework, Radical Pilot (RP), previously called as SAGA Pilot or BigJob, is a powerful means to develop an efficient pipeline tool with distributed resources which is a good example of distributed applications[35, 36]. The core properties of RP are designed for pilot resources and deployment, and workload semantics, binding, and execution, which is altogether capable of effective executions of tasks on heterogeneous distributed resources.

There are advantages with the use of the pilot framework for distributed applications. They include i) that multiple heterogeneous resource utilization and distributed job management are fully and effectively capable, ii) that an extensive solution for

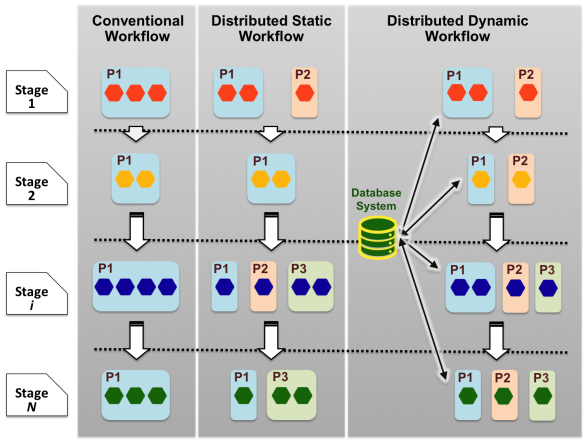


Fig. 2. Three different workflow patterns are illustrated. These workflows represent the execution patterns supported by the Radical Pilot system.

task-level and data-level parallelization for scale-out and scale-across scenarios can be achievable, and iii) that the utilization of accumulated experiences with previous pipeline development projects is rapidly applied for a new target application even requiring specifically customized options. Later, when we detail our main pipeline, it becomes apparent how these advantages are realized with the implementation.

Such major advantages are, in fact, related to the capacity of the pilot paradigm that identifies and support common workflow patterns. As shown in Fig. 2, we consider three scenarios of workflows for pipelines. They are distinctively different in task-resource mapping as well as parallel task execution for each stage of a pipeline. Generally, an entire pipeline for a target application is assumed to be composed of multiple separate stages and each stage needs to execute multiple tasks that are often concurrently conducted. To execute each stage of a pipeline, we run pilots and each pilot, denoted as P1, P2, ... , is responsible for assigned multiple sub-tasks by starting, monitoring, and restarting them. If needed, configurations of pilots could be dynamically made to improve further the computational efficiency. This is feasible since all pilot jobs are controlled and monitored via the back-end database system that updates run-time information on the fly.

Starting with a simple pattern (Conventional Workflow) for which all pilot jobs are executed on a single system, the next pattern is to utilize distributed systems as it is beneficial for decreasing overall Time-To-Completion (TTC) with more resources. The initial choice for the pilot-based workflow with distributed resources is the static workflow in which the distributed job management with pilots are pre-defined (Distributed Static Workflow). This static workflow can be extended to support a dynamically adaptive workflow pattern (Distributed Dynamic Workflow). In this third scenario, the matching of pilot jobs for sub-tasks with computing resources could be decided just before each stage starts. This means that the decision is made with the dynamic information on

TABLE I
De novo assemblers integrated for the RNA-seq pipeline

Name	Type	Distributed Impl.	Version	Ref
Ray	DBG	MPI	2.3.1	[31]
ABySS	DBG	MPI	1.9.0	[39]
Contrail	DBG	Hadoop MapReduce	0.8.2	[24]

computing environment as well as configuration of each stage of the pipeline which is likely to be unknown until the previous stage finishes.

D. Transcript assembly with de novo assemblers and transcript assembly quality evaluation

As summarized In Table I, three de novo assemblers, including the newly supported two tools by us, ABySS and Contrail, were of interest for benchmarking scalable options. Owing to their capabilities for multi-node distributed nothing systems, any size of data sets can be processed. Ray and ABySS are implemented with MPI and Contrail is based on Hadoop MapReduce. In fact, initially, the two other assemblers, MPI-based SWAP[37] and Hadoop-based CloudBrush[38] were also tested, but not included in this work since we found that SWAP was incapable of assemblies with k-mer more than 31 and no support for CloudBrush from developers.

E. Data Sets

In Table II, the datasets we used for the benchmark experiments are summarized. The sizes of these data sets indicate approximately data volumes for a typical prokaryote and a fungal system, respectively. By comparing results with the two data sets, it is also informative for an understanding of characteristic changes in computational requirement as the data size increases. It is worth stating that the size of the P. Crispa sequencing data set is already too large to use c3.2xlarge in which the memory is 16 GB.

When DBG assemblers are utilized, Rnnotator needs multiple k-mer assemblies due to its DBG-based assembly strategy. In our case, the sequencing data with B. Glumae needs 7 assembly calculations with k= 35,37, 39,41,43, 45, and 47, and P. Crispa needs 4 assemblies, i.e. k=51,55,59, and 63. Note that the number of k-mer calculations required is not known until the end of the pre-processing step, consequently implying the need of a dynamical workflow to efficiently manage the number of independent assembly calculations.

IV. Results and Discussion

A. Scalable adaptive pipeline on EC2 for large-scale transcriptome analysis

Here, the three major aspects of the development we aim to achieve with the pipeline are briefed. Overall, the pipeline is built upon a hierarchical architecture; RP provides the low level functionality with its Application Programming Interfaces (APIs), the framework for distributed application built upon RP is developed for transcriptome analysis and reusable for similar execution patterns, and finally the pipeline,

TABLE II
DATA SETS FOR BENCHMARK EXPERIMENTS. THE B. GLUMAE RNA-SEQ DATA IS FROM THE GEO DATABASE (SRA ID: SRX129586). THE DATA FOR P. CRISPA IS FROM THE PREVIOUS STUDY[2]

Organism	B. Glumae	P. Crispa
Description	Bacteria	Fungus
Genome Size	6.7 Mb	34.5 Mb
Number of Protein Genes	5,223	13617
Seq. Data Size (fastq)	3.8 GB	26.2 GB
Read length (bp)	50	100
Num. of reads	16,263,310	54,168,576 x 2
Seq. Platform	Illumina GAII	Illumina HiSeq
Paired end	No	Yes
Memory for Pre-Processing	$\leq$ 15 GB	$\approx$ 40 GB
Data size after pre-processing	175 MB	9.4 GB
k-mer for transcript assembly	35, 37, 39, 41, 43, 45, and 47	51, 55, 59, and 63

built upon the two low level software stacks, is developed for operating a target application of transcriptome profiling with Rnnotator and other available features, over distributed resources (see Fig. 5).

i. Support of parallel executions with the distributed application framework The use of RP for our purpose is primarily useful for the effective optimization of coarse-grained parallelism. By developing the framework, the main goals are twofold; the support of massive parallel tasks and the effective multiple tool integration. Specifically, the transcript assembly step of the Rnnotator workflow is significantly enhanced with these enhancements. By effectively coordinating executions of de novo assemblers capable of running on shared-nothing distributed memory systems with the RP-based framework, any size of large data sets can be processed while enabling a concurrent employment of multiple tools.

ii. Seamless utilization of multiple heterogeneous resources Effectively accessing multiple heterogeneous resources is particularly beneficial for the pipeline development including the support of scale-across that increases the scalability over multiple computing resources. Facilitated by this feature, the four different stages in our pipeline can be executed in different computing systems or on multiple machines concurrently. In coming years, more extreme-scale tasks of the transcript assembly need to be conducted, and our pipeline is intrinsically capable with a little amount of changes. In a single cluster system, like HPC, the scale-out execution is mostly made with a local scheduler such as SGE or PBS. RP provides an easy way to work with such schedulers. Since the original architecture for Rnnotator is designed for such HPC environments, our pipeline creates a cluster using multiple VMs and thus MPI-based or Hadoop-based applications are executed with such a local scheduler. We utilize StarCluster[40] for creating a cluster system that contains SGE. Since the available Amazon Machine Images (AMI) of StarCluster is not compatible with the latest Ubuntu required by other software tools for our purpose for the pipeline, we created a new customized StarCluster script.

iii. Support of dynamic workflow The static workflow-based implementation was developed initially and then a further optimization for a dynamic scheme was attempted. The fully dynamically adaptive workflow is not implemented yet, however, at this time. Rather, we describe the current development efforts. Specifically, as an example of dynamically adaptive schemes, the pilot-based transcript assembly is implemented, for which the information retrieved from the output of the pre-processing step is utilized.

TABLE III
BASELINE PERFORMANCE OF THE THREE DE NOVO ASSEMBLERS FOR TRANSCRIPT ASSEMBLY. TIME-TO-COMPLETION IS MEASURED WITH A TWO-NODE CLUSTER USING B. GLUMAE DATA. K-MER SIZE IS 47. EC2 INSTANCE TYPE USED IS C3.2XLARGE.

Assembler	TTC (sec)
Ray	1,721
ABySS	882
Contrail	6,720

TABLE IV
THE TWO INSTANCE TYPES, C3.2XLARGE AND R3.2XLARGE, ARE COMPARED WITH RESPECT TO THEIR CAPACITY FOR THE TWO DIFFERENT DATA SETS. X MEANS "NOT SUPPORTED".

Task	Dataset	c3.2xlarge	r3.2xlarge
Pre-Processing	B. Glumae	O	O
	P.Crispa	X	O
Transcript Assembly with Ray	B. Glumae	O	O
	P. Crispa	X	O
Transcript Assembly with ABySS	B. Glumae	O	O
	P. Crispa	X	O
Transcript Assembly with Contrail	B. Glumae	O	O
	P. Crispa	X	O
Post-Processing	B. Glumae	O	O
	P. Crispa	O	O

B. Benchmark experiments

First of all, the baseline performance of the de novo assemblers is measured and compared in Table III. This can be useful information for evaluating the performance gain with scalable solutions based on the utilization of distributed resources. Starting with this reference performance, benchmark results, as hinted in Table II, emphasize required changes in computational costs and other requirements as sequencing data volume, the required number of k-mer assembly, and therefore optimized conditions with the three assemblers become different. For example, as highlighted in Table IV, the bigger data set of P. Crispa suffers a failure with less powerful instance types.

Our benchmark results also contain an example of potential benefits with an efficient multiple tool integration. Our new pipeline can carry out the transcript assembly with the available assemblers, separately or together. We present a simple comparative study on the transcript assembly quality for those cases. To this end, we utilize DETONATE[41].

i. Scale-out performance of assemblers
An understanding of the potential performance of de novo assemblers implemented with MPI or Hadoop MapReduce with respect to scale-out is important for searching optimal options for the pipeline and thus supported by the pilot framework. First of all, three de novo assemblers were observed

to show different performance in scale-out conditions. Note that our comparison is for RNA-seq data sets, and thus could be different from other prior works focusing on genome assembly. In Fig. 3, results with the P. Crispa data set is shown. Results with B. Glumae was found to show a similar pattern (unpublished). For this, the original RNA-seq data sets were provided as input without pre-processing. The exception is the P. Crispa data set with Contrail for which the pre-processed data are used in order to avoid the failure due to the reads containing nucleotides with N.

Noticeably, Contrail is very slow and inefficient until the sufficient number of nodes are used (see also Table III). This is understandable since Hadoop-based tools are primarily favored for large-scale distributed tasks and not optimized with a small number of workers . When more nodes are added, TTC is becoming close. Regarding the scale-out performance, on the other hand, when additional nodes are utilized, ABySS does not show any significant gain in TTC compared to Ray showing a marginal gain.

Therefore, the main reason for the use of two MPI-based assemblers for large sequencing data should be because of the total distributed memory increased for more data sets, not because of the scalability. The observed scalability among MPI-based tools is less encouraging, indicating that, in spite of studies showing the notable scalability of MPI-based assemblers[37], the difficulty of such implementations seems to be non-trivial. Interestingly, we believe that Hadoop-based approaches still have potentials. For example, it is intriguing to see whether the better scalability, in spite of the relatively low performance with the MapReduce-based tool, can be achieved if other programming models and software stacks such as SPARK, HAMA, and many others are employed. This is because it is now well-known that MapReduce is less effective for iterations of distributed parallel in-memory tasks, which is in fact the case with the transcript assembly.

ii. Task-level parallelization for multiple k-mer assembly

In addition to the scale-out performance of individual assemblers, an understanding of possible options for parallel tasks required for the transcript assembly step is crucial for the pipeline. In Fig. 4, the performance of the transcript assembly step is investigated using the assembly with Ray. The data set is from P. Crispa, but we used a partial data set due to the computational cost with the entire data set. The number of k-mer calculations needed are 4, and these 4 different tasks for which each task corresponding to a single k-mer assembly show the similar scale-out behavior as already seen with unprocessed data sets in Fig. 3. Here, we confirm that such a behavior is uniformly expected regardless of the data size. Results shown in the low panel of Fig. 4, shed lights on another aspect that additional gains could be obtained if an efficient task management scheme is supported for parallelizing the number of assembly tasks. Interestingly, it is found that the assembly with 3 nodes (24 cores) still shows a slight gain from a case with 2 nodes, indicating the benefits using more nodes in TTC. This is basically the optimization problem for heterogeneous tasks arising from different k-mer assemblies

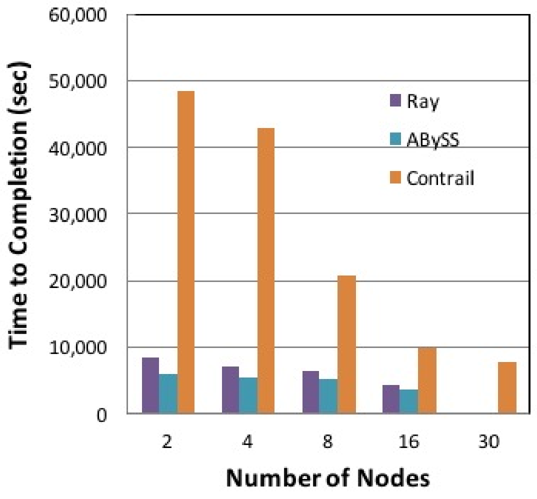


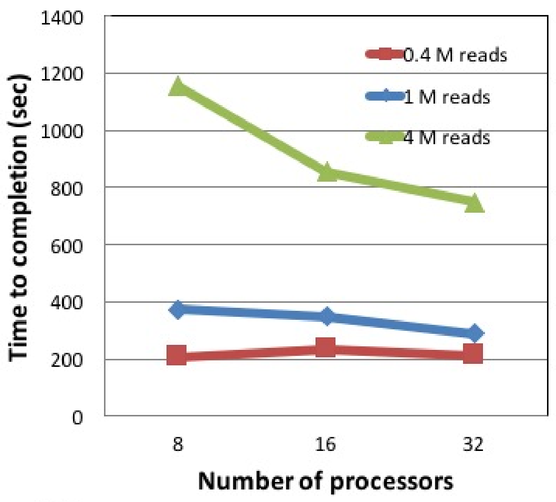
Fig. 3. Scale-out performance of the three assemblers that are integrated for transcript assembly. The data set of P. Crispa is used. The EC2 instance type used is c3.2xlarge that has 8 cores in a single node. k-mer size is set to 51.

as well as assembly tasks with multiple assemblers. Note that the pilot framework was utilized in our previous work for a similar goal on EC2[42]. More complicated situations were also examined for influential factors deciding TTCs or expenses. Examples include the number of nodes for each MPI job vs. the number of k-mer assemblies, but not presented here due to its complexity for the interpretation.

iii. Transcript assembly with multiple assemblers

Since our pilot-based pipeline is effectively scalable for running multiple assemblers for the transcript assembly, we evaluate the accuracy of results using multiple options corresponding to a single assembler or a combination of assemblers. The latter approach that combines results from multiple assemblers is indeed the Multi-assembler Multi-parameter (MAMP) method. For an evaluation of transcript assembly, we used the scores suggested from the tool, DETONATE[41]. These metrics are recall, precision, and F_1 values calculated in the nucleotide level, and the weighted k-mer recall and the kc score among those from the reference-based measures of DETONATE. The RNA-seq data from B. Glumae was used for this comparison and the reference transcript sequences used as the ground truth are 6234 gene sequences from the NCBI GenBank database (http://ncbi.nlm.nih.gov). Note that our results do not necessarily represent the real transcript quality accurately, due to multiple factors. For example, the ground truth sequences are not the entire mRNA transcripts, rather they are protein gene sequences predicted by the annotation programs using the whole genome sequences. Therefore, our results should be considered as an initial attempt to explore potential benefits and future directions for major objectives of the pipeline .

Nonetheless, the results summarized in Table V suggest many intriguing findings. First of all, all transcript assembly



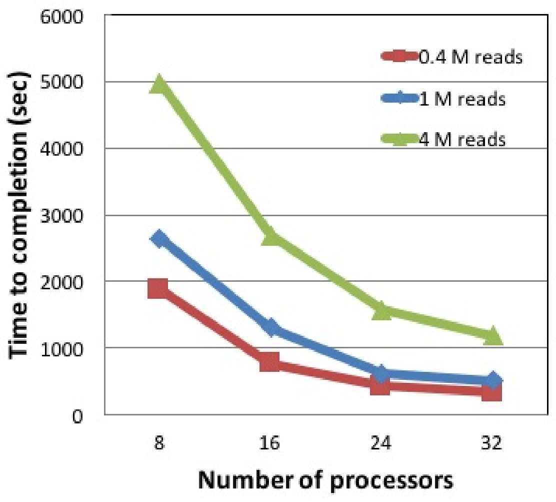


Fig. 4. In the upper panel, the scalability of Ray assembler is shown with respect to the size of the input and the number of cores. In the lower panel, the scalability of the assembly step using Ray that requires to run multiple k-mer calculation is shown. These results collectively indicate that there exist the two different types of parallelisms for sub-tasks in the transcript assembly step. For these benchmark results, the instance used is r3.2xlarge offered with 8 cores.

results with our pipeline, regardless of different options, are better than (w.r.t. nucleotide level) or comparable to (weighted score) the result with Trinity, one of the most popular programs. Note that the pre-processing step of Trinity is different from our pipeline, and thus the direct comparison needs to be scrutinized. Nonetheless, the results indicate the robust performance of the main workflow adopted from Rnnotator. The favorable performance is further indicated by the improved results for recall when the weighted scheme is used. According to DETONATE, the weighted scheme considers the abundance of reads supporting assembled transcript sequences, and thus increased recall values suggest a good quality of transcripts for cases strongly supported by read data.

TABLE V
COMPARISON OF TRANSCRIPT ASSEMBLY QUALITY. THE DATA SET OF B. GLUMAE IS USED. THE SCORE CALCULATION IS OBTAINED USING REFERENCE-BASED METRICS IN DETONATE V1.10. FOR THE COMPARISON, THE EVALUATION RESULT OF ASSEMBLY CONTIGS OBTAINED WITH TRINITY[3] (V 2.1.1) IS ALSO SHOWN.

Assembler Used	Nucleotide-level (precision, recall, F_1)	(weighted k-mer recall, kc score)
Ray	0.84, 0.26, 0.40	0.86, 0.86
ABySS	0.82, 0.42, 0.55	0.79, 0.78
Contrail	0.78, 0.43, 0.56	0.84, 0.83
Ray + Contrail	0.78, 0.43, 0.56	0.78, 0.77
Ray+Contrail+ABySS	0.79, 0.44, 0.57	0.77, 0.76
Trinity	0.51, 0.35, 0.42	0.84, 0.83

Importantly, the scalable capacity of our pipeline that can carry out multiple options at the same time allows end users to simply choose their best option if he/she can use an independent metric for the comparison. On the other hand, the two options of the MAMP strategy are not apparently better than options with a single assembler, even though their performance seems to be optimal to the average values. In spite of this initial result, our exploration favoring this kind of ensemble approaches is encouraged by the success of ensemble methods which are fairly well known in the fields of statistical learning and inference. In many cases, they were shown to outperform other approaches utilizing a single model or classifier[16]. It is worth stating that our current implementation for merging the information of contigs generated from multiple assemblers employs the default setting of Rnnotator. While this default approach is thought to be appropriate for merging multiple k-mer assemblies with a single assembler, there seems to be higher opportunities to show better performing MAMP-based methods in the future with novel ideas for validating transcripts and properly merging them.

C. Toward dynamic adaptive workflow

The ultimate goal of our project with the pipeline is to develop the public research resource supporting a fully dynamically adaptive workflow, which will be served via a web-based science gateway to the research community. To this end, we need the logic with which a workflow is created and executed in an efficiently adaptive manner reflecting dynamically changing environmental conditions as well as parameters generated on the fly. As proposed in the previous work[42], for such a goal, factors and conditions affecting the performance of a workflow should be known, along with a means for a rough estimate on TTCs of sub tasks a priori. In this work, our focus is to understand such aspects, particularly, by exploiting benchmark experiments primarily designed for such purposes.

Here, we describe the entire workflow in details, highlighting our current development level toward the goal. At the end, to offer a closer look from the end user perspective, a sample run of the pipeline, using the B.Glumae data set and the option for three assemblers together in the transcript assembly, is presented. For this sample run, a reasonably pre-

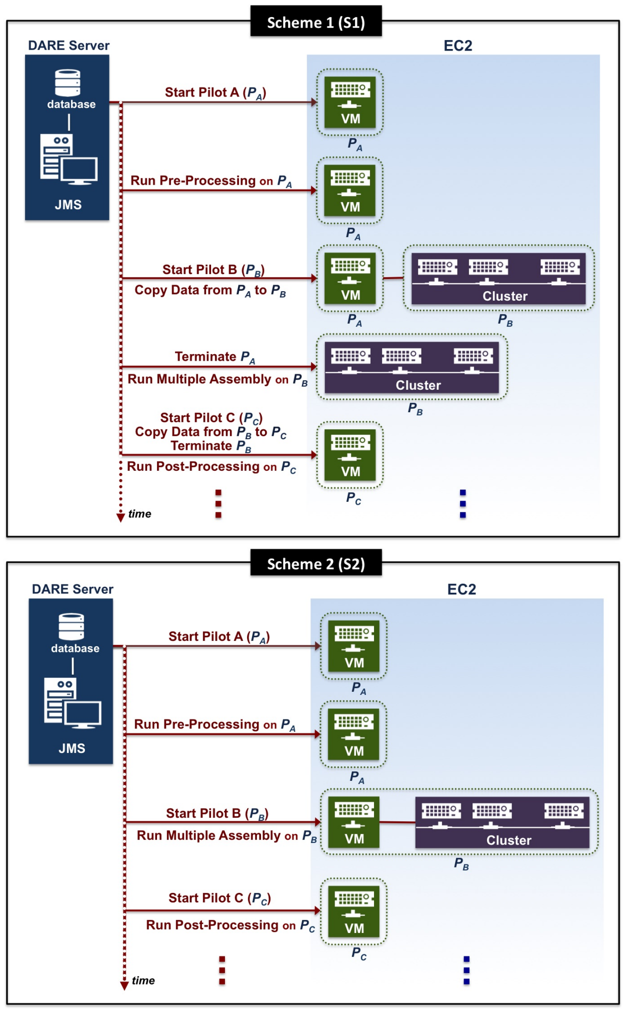


Fig. 5. The two schemes (S1 and S2) for on-demand computing environments like EC2 are illustrated. They differ in how to match a pilot with corresponding VMs. Details are explained in the text. The overall system architecture serving a pipeline is also schematically shown. JMS represents Job Management System which submits and orchestrates a workflow.

defined configuration is used and observed cost and TTC are reported.

First of all, an entire workflow needs to choose one of two different options in the beginning. The main reason for the need of such options is because of the unique computing environment of on-demand clouds. Unlike a conventional HPC environment, on-demand computing clouds require a user to choose types of instances and to be responsible for starting and stopping VMs. In order to deal with such a situation, we decided to support the pilot-matching schemes in two different ways as shown in Fig. 5. The first option, the matching scheme 1 (S1), couples a pilot and the lifetime of VM such that a new pilot always starts with the creation of VMs needed and ends with the termination of VMs when the role of the pilot finishes. On the other hand, the other option, denoted as the matching scheme 2 (S2), allows to reuse currently running VMs for a new pilot, resulting in a decoupling mechanism between a pilot and a lifespan of VMs for it.

S2 represents somehow a common scenario in traditional HPC environments, and thus conveniently reusable for the future extension of our pipeline incorporating distributed HPC resources. S1 is notably beneficial for an optimal execution since appropriate resource types can be chosen among EC2 instances for each pilot, lowering TTC or cost depending upon the priority goal of the target execution. However, it cannot avoid overheads stemming from extra tasks for starting and terminating VMs as well as the data transfer between VMs of newly created and those that are going to be terminated. On the contrary, S2 has no such overheads, but is likely to be less efficient in cases when the mandatory reuse of existing VMs for the next step constrains a better utilization of resources. For example, the large input size of the P. Crispa data set prohibits the pre-processing step with an instance equipped with less than 40 GB memory, implying r3.2xlarge should be used. This ends up with forcing the transcript step to keep unnecessarily this expensive instance.

Once the decision between S1 and S2 is made, the pre-processing step is started with the pilot P_A. While the sizes of two data sets for this work do not require changes in the original implementation of Rnnotator, in order to deal with much bigger data sets, the future implementation needs to support distributed data-level parallelization for this step. Currently, depending upon the size of input data, an appropriate type of instance equipped with sufficiently large memory is chosen. c3.2xlarge is fine with B. Glumae but inappropriate for P. Crispa (see Table IV).

After pre-processing, P_B, starts with the required number of VMs for the transcript assembly. The two parameters such as the number of nodes for each k-mer and the required number of k-mer assemblies, need to be decided before P_B. Note that the latter parameter should be known by using the information obtained from the pre-processing step. As already shown with the benchmark results, each assembler behaves with the different scalability for each k-mer and the overall performance varies with different configurations of parallel execution of multiple k-mer assemblies. Considering these factors, the optimal number of nodes could be found. For the cluster set up in this step, our customized StarCluster using version 0.95.6 was used. MPI jobs for ABySS and Ray, or Hadoop jobs for Contrail are submitted to Sun Grid Engine (SGE) scheduler available with the StarCluster AMI. For the multiple assembler options, it is possible to submit multiple k-mer jobs for each assembler together to SGE or separately. Finally, the following post-processing step and the step for gene expression level calculation (with optional differential gene expression) are carried out by P_C using a single VM. In general, the data size for these steps is a lot less than the original sequencing read data, and thus a single VM is fine for our data sets. In this step, our new developmental contribution for the support of multi assembler options is implemented in the post-processing task.

In the following, the example run is summarized, along with specific conditions and configurations chosen for the multiple assembly option. The matching scheme, S2, is chosen and c3.2xlarge is the choice for the pre-processing as well as for all following steps. We used the unpublished real sequencing data set for B. Glumae. The used data set is paired-end and the total size is 4.4 GB. The k-mers needed for the transcript assembly are 2. Once one VM was launched for pre-processing as P_A, the input data is sent from the local server to the VM, taking about 3 min 35 sec. The pre-processing step takes 44 min. After pre-processing, 35 VMs are needed to be created additionally, resulting in a cluster of 36 nodes, which belongs to P_B. The total 6 jobs, corresponding to two k-mer assemblies for each assembler, are submitted to the SGE scheduler of the created cluster. This configuration corresponds to 4 MPI jobs for Ray and ABySS and 2 MapReduce-based Contrail jobs. MPI jobs are configured to run on a single node with 8 slots, and MapReduce-based jobs use 16 nodes. This decision is based on the preliminary benchmark finding that there is no significant benefit with MPI jobs with more than a single node for the two MPI assemblers, whereas the results with Contrail suggest that at least 16 nodes are needed to match TTCs of the MPI assemblers in the case of B. Glumae. Overall, the assembly step with P_B takes 1 hour 18 min. Note that this TTC is in fact the longest one required for the Contrail-based assembly of two k-mer calculations. For tasks of Contrail, 1 min is additionally needed for the file format conversion to SFA from Fastq. After this step, the post-processing step starts with a single VM, corresponding to P_C. Again, this VM is the one having existed from the beginning (i.e. the same resource for P_A and one of nodes for P_B). Other 35 VMs, which are not necessary for P_C, are terminated. P_C takes 41 min. Again, the overall workflow has no need of file movement among resources allocated for different pilots since the same VM serves for all three pilots. Overall, we can finish this run in 2 hours 47 min, and the cost is about 20.28 USD.

V. Future works and concluding remarks

Genome-wide transcriptome analysis is a complex process. The ongoing revolution in sequencing technology deepens the gap between analysis tasks and ever-growing data. This gap is more widen as many outstanding roadblocks are likely to arise from new progresses in metatranscriptome, use cases of multi-platform methods, single cell sequencing, and more sequencing data with non-model species. This strongly suggests the need of an integrative tool that can support massive data processing as well as computational tasks in an efficient way.

In this work, we present our developmental outcomes to address outstanding challenges toward such an integrative tool. Based on the Rnnotator pipeline tool, our strategy is to provide the scalability framework using the pilot system, Radical Pilot. Specifically, as demonstrated with examples, the use of de novo assemblers implemented with MPI or Hadoop resolves the immediate concern to deal with bigger data sets without a failure. Additionally, the support of multiple assembler options is not only useful to find the best one among different results, but also is an attractive platform for more advanced methods based on ensemble and Multi-assembler Multi-parameter (MAMP) methods without worrying a required computational burden.

Continuing our effort to enhance the capacity of the pipeline, the following directions are prioritized. Firstly, other steps such as pre-processing and post-processing are also to be more pilot-powered by supporting efficient data and task-level parallelization over distributed systems. Secondly, the main component for driving the dynamic adaptive workflow will be implemented. Thirdly, the pipeline will be fully tested for OpenStack. Eventually, it is possible to support the scale-across execution of Rnnotator that supports multiple heterogeneous distributed computing resources comprising of HPC systems and on-demand computing clouds. Other directions include algorithmic development, for example, such as new implementations for better transcript assembly using an ensemble-based method. Finally, the pipeline will be soon available to the research community via the science gateway project (http:dare.cct.lsu.edu).

Acknowledgment

We are thankful for Amazon EC2 computing time with the AWS research grant program. We thank Colin Dewey and his group member for helping us for the use of DETONATE. This research was supported in part by the funding from NIH P20 GM103458-10.

References

[1] Z. Wang, M. Gerstein, and M. Snyder. RNA-seq : a revolutionary tool for transcriptomics. *Nat. Rev. Genet.*, 10(1):57–63, 2009.

[2] Sean Gordon, Elizabeth Tseng, Asaf Salamov, Jiwei Zhang, Xiandong Meng, Zhiying Zhao, Dongwan Don Kang, Jason Underwood, Igor V Grigoriev, Melania Figueroa, et al. Widespread polycistronic transcripts in mushroom-forming fungi revealed by Single-Molecule long-read mRNA sequencing. *PLoS One*, 10(7): e0132628, 2015.

[3] Manfred G Grabherr, Brian J Haas, Moran Yassour, Joshua Z Levin, Dawn A Thompson, Ido Amit, Xian Adiconis, Lin Fan, Raktima Raychowdhury, Qiandong Zeng, et al. Full-length transcriptome assembly from RNA-Seq data without a reference genome. *Nature biotechnology*, 29(7):644–652, 2011.

[4] Zheng Chang, Zhenjia Wang, and Guojun Li. The impacts of read length and transcriptome complexity for de novo assembly: A simulation study. *PloS One*, 9:e94825, 2014.

[5] Jeffrey Martin, Vincent M Bruno, Zhide Fang, Xiandong Meng, Matthew Blow, Tao Zhang, Gavin Sherlock, Michael Snyder, and Zhong Wang. Rnnotator: an automated de novo transcriptome assembly pipeline from stranded RNA-Seq reads. *BMC genomics*, 11(1):663, 2010.

[6] Malachi Griffith, Jason R Walker, Nicholas C Spies, Benjamin J Ainscough, and Obi L Griffith. Informatics for RNA sequencing: A web resource for analysis on the cloud. *PLoS Comput Biol*, 11(8):e1004393, 2015.

[7] Andre Merzky, Mark Santcroos, Matteo Turilli, and Shantenu Jha. Radical-Pilot: Scalable execution of heterogeneous and dynamic workloads on supercomputers, 2015. http://arxiv.org/abs/1512.08194.

[8] Joohyun Kim, Sharath Maddineni, and Shantenu Jha. Characterizing deep sequencing analytics using bfast: Towards a scalable distributed architecture for next-generation sequencing data. In

Proceedings of the Second International Workshop on Emerging Computational Methods for the Life Sciences, ECMLS '11, pages 23–32, New York, NY, USA, 2011. ACM.

[9] Joohyun Kim, Sharath Maddineni, and Shantenu Jha. Advancing next-generation sequencing data analytics with scalable distributed infrastructure. *Concurrency and Computation: Practice and Experience*, 26(4):894–906, 2014.

[10] Soon-Heum Ko, Nayong Kim, Joohyun Kim, Abhinav Thota, and Shantenu Jha. Efficient runtime environment for coupled multi-physics simulations: Dynamic resource allocation and load-balancing. In *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on*, pages 349–358. IEEE, 2010.

[11] Dent Earl, Keith Bradnam, John St John, Aaron Darling, Dawei Lin, Joseph Fass, Hung On Ken Yu, Vince Buffalo, Daniel R Zerbino, Mark Diekhans, et al. Assemblathon 1: a competitive assessment of de novo short read assembly methods. *Genome research*, 21(12):2224–2241, 2011.

[12] Keith R Bradnam, Joseph N Fass, Anton Alexandrov, Paul Baranay, Michael Bechner, Inanç Birol, Sébastien Boisvert, Jarrod A Chapman, Guillaume Chapuis, Rayan Chikhi, et al. Assemblathon 2: evaluating de novo methods of genome assembly in three vertebrate species. *GigaScience*, 2(1):1–31, 2013.

[13] Sergey Koren, Todd J Treangen, Christopher M Hill, Mihai Pop, and Adam M Phillippy. Automated ensemble assembly and validation of microbial genomes. *BMC Bioinformatics*, 15(1):126, 2014.

[14] Xutao Deng, Samia N Naccache, Terry Ng, Scot Federman, Linlin Li, Charles Y Chiu, and Eric L Delwart. An ensemble strategy that significantly improves de novo assembly of microbial genomes from metagenomic next-generation sequencing data. *Nucleic Acids Research*, 43(7):e46–e46, 2015.

[15] Joanna Moreton, Stephen P Dunham, and Richard D Emes. A consensus approach to vertebrate de novo transcriptome assembly from RNA-seq data: assembly of the duck (Anas platyrhynchos) transcriptome. *Frontiers in Genetics*, 5, 2014.

[16] Pengyi Yang, Yee Hwa Yang, Bing B Zhou, and Albert Y Zomaya. A review of ensemble methods in bioinformatics. *Current Bioinformatics*, 5(4):296–308, 2010.

[17] B. Langmead and et al. Cloud-scale RNA-sequencing differential expression analysis with Myrna. *Genome Biol.*, 11(8):R83, 2010.

[18] Dongwan Hong, Arang Rhie, Sung-Soo Park, Jongkeun Lee, Young Seok Ju, Sujung Kim, Saet-Byeol Yu, Thomas Bleazard, Hyun-Seok Park, Hwanseok Rhee, et al. FX: an RNA-Seq analysis tool on the cloud. *Bioinformatics*, 28(5):721–723, 2012.

[19] M D'Antonio, P DM D'Onorio, M Pallocca, E Picardi, AM D'Erchia, R Calogero, T Castrignanò, and G Pesole. RAP: RNA-Seq Analysis Pipeline, a new cloud-based NGS web application. *BMC Genomics*, 16(Suppl 6):S3, 2015.

[20] Shanrong Zhao, Kurt Prenger, and Lance Smith. Stormbow: a cloud-based tool for reads mapping and expression quantification in large-scale RNA-Seq studies. *ISRN bioinformatics*, 2013, 2013.

[21] R. C. Taylor. An overview of the Hadoop/MapReduce/HBase framework and its current applications in bioinformatics. *BMC Bioinformatics*, 11:S1, 2010.

[22] Christopher Moretti, Andrew Thrasher, Li Yu, Michael Olson, Scott Emrich, and Douglas Thain. A framework for scalable genome assembly on clusters, clouds, and grids. *Parallel and Distributed Systems, IEEE Transactions on*, 23(12):2189–2197, 2012.

[23] A. McKenna, M. Hanna, E. Banks, and et al. analyzing next-generation DNA sequencing dataThe Genome Analysis Toolkit: A MapReduce framework for . *Genome Res.*, 20:1297–1303, 2010.

[24] Michael Schatz, Dan Sommer, David Kelley, and Mihai Pop. Contrail: Assembly of large genomes using cloud computing. In *CSHL Biology of Genomes Conference*, 2010.

[25] Marek S Wiewiórka, Antonio Messina, Alicja Pacholewska, Sergio Maffioletti, Piotr Gawrysiak, and Michał J Okoniewski. Sparkseq: fast, scalable, cloud-ready tool for the interactive genomic data analysis with nucleotide precision. *Bioinformatics*, 30(18):2652-2653, 2014.

[26] Spark. http://spark.apache.org.

[27] Stefan Kurtz. The vmatch large scale sequence analysis software. *Ref Type: Computer Program*, pages 4–12, 2003.

[28] Daniel D Sommer, Arthur L Delcher, Steven L Salzberg, and Mihai Pop. Minimus: a fast, lightweight genome assembler. *BMC Bioinformatics*, 8(1):64, 2007.

[29] Daniel R Zerbino and Ewan Birney. Velvet: algorithms for de novo short read assembly using de bruijn graphs. *Genome research*, 18(5):821–829, 2008.

[30] Marcel H Schulz, Daniel R Zerbino, Martin Vingron, and Ewan Birney. Oases: robust de novo RNA-seq assembly across the dynamic range of expression levels. *Bioinformatics*, 28(8):1086–1092, 2012.

[31] Sébastien Boisvert, François Laviolette, and Jacques Corbeil. Ray: simultaneous assembly of reads from a mix of high-throughput sequencing technologies. *Journal of Computational Biology*, 17(11):1519–1533, 2010.

[32] Yu Peng, Henry CM Leung, Siu-Ming Yiu, and Francis YL Chin. Idba–a practical iterative de bruijn graph de novo assembler. In *Research in Computational Molecular Biology*, pages 426–440. Springer, 2010.

[33] Rayan Chikhi, and Guillaume Rizk. Space-efficient and exact de Bruijn graph representation based on a Bloom filter *Algorithms for Molecular Biology*, 8(1):1-9, 2013.

[34] Sharath Maddineni, Joohyun Kim, Yaakoub El-Khamra, and Shantenu Jha. Distributed Application Runtime Environment (DARE): A Standards-based Middleware Framework for Science-Gateways *Journal of Grid Computing*, 10(4):647-664, 2012.

[35] Radical Pilot. http://radical-cybertools.github.io/radical-pilot/index.html.

[36] Matteo Turilli, Mark Santcroos, and Shantenu Jha. A Comprehensive Perspective on Pilot-Abstraction, 2015. http://arxiv.org/abs/1508.04180.

[37] Jintao Meng, Bingqiang Wang, Yanjie Wei, Shengzhong Feng, and Pavan Balaji. Swap-assembler: scalable and efficient genome assembly towards thousands of cores. *BMC Bioinformatics*, 15(Suppl 9):S2, 2014.

[38] Yu-Jung Chang, Chien-Chih Chen, Chuen-Liang Chen, and Jan-Ming Ho. A de novo next generation genomic sequence assembler based on string graph and mapreduce cloud computing framework. *BMC Genomics*, 13(Suppl 7):S28, 2012.

[39] Jared T Simpson, Kim Wong, Shaun D Jackman, Jacqueline E Schein, Steven JM Jones, and Inanç Birol. Abyss: a parallel assembler for short read sequence data. *Genome Research*, 19(6):1117–1123, 2009.

[40] StarCluster. http://star.mit.edu/cluster/.

[41] Bo Li, Nathanael Fillmore, Yongsheng Bai, Mike Collins, James A Thomson, Ron Stewart, and Colin N Dewey. Evaluation of de novo transcriptome assemblies from RNA-Seq data. *Genome Biology*, 15(12):553, 2014.

[42] Anjani Ragothaman, Sairam Chowdary Boddu, Nayong Kim, Wei Feinstein, Michal Brylinski, Shantenu Jha, and Joohyun Kim. Developing eThread Pipeline Using SAGA- Pilot Abstraction for Large-Scale Structural Bioinformatics. *BioMed Research International*, 2014:348725, 2014.