Understanding MapReduce-based Next-Generation Sequencing Alignment on Distributed Cyberinfrastructure

Pradeep Kumar Mantha
Center for Computation and Technology
Louisiana State University
216 Johnston
Baton Rouge, LA
pmanth2@cct.lsu.edu

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

Andre Luckow
Center for Computation and Technology
Louisiana State University
216 Johnston
Baton Rouge, LA
aluckow@cct.lsu.edu

Joohyun Kim
Center for Computation and Technology
Louisiana State University
216 Johnston
Baton Rouge, LA
jhkim@cct.lsu.edu

Shantenu Jha
Center for Autonomic Computing
Rutgers University
94 Brett Road
Piscataway, NJ
shantenu.jha@rutgers.edu

ABSTRACT

Although localization of Next-Generation Sequencing (NGS) data is suitable for many analysis and usage scenarios, it is not universally desirable, nor possible. However most solutions "impose" the localization of data as a pre-condition for NGS analytics. We analyze several existing tools and techniques that use MapReduce programming model for NGS data analysis to determine their effectiveness and extensibility to support distributed data scenarios. We find limitations at multiple levels. To overcome these limitations, we developed a Pilot-based MapReduce (PMR) – which is a novel implementation of MapReduce using a Pilot task and data management implementation. PMR provides an effective means by which a variety of new or existing methods for NGS and downstream analysis can be carried out whilst providing efficiency and scalability across multiple clusters. Pilot-MapReduce (PMR) circumvents the use of global reduce and yet provides an effective, scalable and distributed solution for MapReduce programming model. We compare and contrast the PMR approach to similar capabilities of Seqal and Crossbow, two other tools which are based on conventional Hadoop-based MapReduce for NGS reads alignment and duplicate read removal or SNP finding, respectively. We find that PMR is a viable tool to support distributed NGS analytics, particularly providing a framework that supports parallelism at multiple levels.

Categories and Subject Descriptors

D.1.3 [**Software**]: Concurrent Programming-Distributed programming/parallel programming; J.3 [**Computer Applications**]: LIFE AND MEDICAL SCIENCES-Biology and genetics

General Terms

Design, Experimentation, Performance

Keywords

Genome Sequence Alignment, BWA, Bowtie, Human Genome, RNA-Seq Data, MapReduce, Distributed Computing, Simple API for Grid Applications (SAGA), Pilot Job and Data

1. INTRODUCTION

Recent advances in high-throughput DNA sequencing technologies such as Next-Generation Sequencing (NGS) platforms have resulted in unprecedented challenges in the areas of bioinformatics and computational biology [1–5]. These challenges are to some extent novel because of the need of the cross-cutting and integrated solutions leveraging algorithmic advances, tools and services, and scalable cyberinfrastructure and middleware.

For dealing with unprecedented data and required data analytics and downstream analyses of such high-throughput deep sequencing techniques, new strategies such as MapReduce-based approaches have been added to an arsenal of computational biologists [6]. Several tools using MapReduce have been already introduced for NGS data analysis such as read alignment onto a reference genome [7–12].

Since the paper about the MapReduce programming model from Google in 2004 [13] and the emergence of Hadoop, the model has drawn significant attention from various application domains as a general strategy for parallelization of processing a large data. Not surprisingly, most of software tools using MapReduce for NGS data were developed in the Hadoop environment and consequently with Cloud computing environment such as Amazon EC2, consistent with the belief that MapReduce implementations on Cloud environments provide an efficient solution for big data problems [13, 6, 12].

Whereas the combination of MapReduce and Clouds works well for scenarios where all "the data can be poured into the Cloud", often this is not possible. It is likely that the proportion/instances where such stringent models of data localization are not possible will increase. Some contributing factors will be: increasing volumes of data and thus greater challenge in transferring & localizing it; increasing number of data-repositories that will need to be accessed – each with their own data-transfer and access policies; greater variety and distribution in the number of producers and consumers of this data.

A consequence of the increasing importance of distributed data in NGS analytics, is that current programming models – or at least programming models as currently implemented and executed, will prove inadequate when faced with the challenge of distributed data; this will be true of analytical approaches and tools that depend upon these programming models [14]. We thus posit that there is an emerging downstream (analysis) problem in the field of NGS.

While there have been algorithmic advances and a plethora of new software tools providing user-friendly interface via web-based tools, client GUI, or integrative workbench environments [15], the end-to-end development of scalable infrastructure is lagging. In other words, it is still commonly observed that many biologists are puzzled by the proliferation of tools, and algorithms, whilst very few tools or solutions exist that are extensible and flexible so as to yield integrated solutions.

The primary objective of our work is to demonstrate that existing approaches can be extended to support distributed data scenarios with simple but effective changes in their execution and runtime environments.

Specifically, we demonstrate the use of a Pilot-based SAGA-MapReduce with which NGS read alignments and subsequent analysis can be conducted in a distributed, yet scalable fashion. Such Pilot-based SAGA-MapReduce, herein referred to as PMR, can be thought of as the capability resulting from writing MapReduce using SAGA [16] to support task and data distribution, whilst exploiting the capabilities of a generalized and extensible Pilot-Job (and data) as a runtime environment[17–19].

Before we discuss further contribution and outline of the paper, we clarify our usage of the different types of scaling that this paper will be concerned with: *scale-up* – or the most common type of scaling behavior, is a reference to the ability (performance) of using many cores efficiently; *scale-out* is a measure of the number of tasks that can be concurrently executed & managed; *scale-across* is a measure of the number of distinct compute resources that an application can utilize.

Our experiments compare and contrast PMR with two known MapReduce-based tools: Seqal in the Seal package and Crossbow [10, 11], for some features, including but not limited to, i) scalability across multiple heterogeneous infrastructures, ii) framework supporting extensions, iii) effectiveness for data-intensive problem with distributed data.

The implementation of the MapReduce programming model using Pilot capabilities in turn supports the development of applications which have significant flexibility in scheduling as well as the ability to exploit resources that are determined at runtime. These features (arising from the Pilot-abstraction) when coupled with interoperability (arising from SAGA), enable such *dynamic applications*, to be executed across a range of distributed resources such as clusters, or a mix of HPC grids and Clouds.

Indeed, based upon points (i) to (iii), PMR provides a viable compute and data management environment for high-throughput DNA sequencing data whilst using distributed cyberinfrastructure effectively. We shall establish that in addition to providing solutions for challenges arising from the required data volume scalability and efficiency, PMR can be a valuable approach for a broad range of NGS tools as well as the emergence of revolutionary NGS techniques such as RNA-Seq. It is instructive to contrast the flexible and extensible support for such tools to other approaches using Hadoop-based MapReduce implementations [7, 9–11].

Unfortunately, users cannot directly deploy a MapReduce framework such as Hadoop on top of multiple clusters to form a single larger MapReduce cluster. Typically the internal nodes of a cluster are not directly reachable from outside. However, MapReduce requires the master node to directly communicate with any slave node, which is also one of the reasons why MapReduce frameworks are usually deployed within a single cluster. Therefore, one challenge is to make multiple clusters act collaboratively as one so it can more efficiently run MapReduce [20].

Hierarchical MapReduce as proposed in [20] is often a viable solution to the distributed data problem, but developing a global reduce is not easy and straightforward in all cases. Indeed, many analyses for NGS data are not appropriate for hierarchical MapReduce. On the contrary, PMR [18] is a distributed solution and circumvents the use of a global reduce, providing an efficient and scalable solution for MapReduce applications.

The paper is organized as follows. In Section 2, the data sets we used for this work and experimental methods are described. After that, in Section 3, our results for parallel performance measurements and comparison of PMR with Seqal and Crossbow which represent applications that use the open source Hadoop-based MapReduce [21, 12, 22, 10] are presented. Finally, in Section 4, we discuss and conclude this paper with remarks focusing on implications of our work for the development of biological information infrastructure around NGS data.

2. MATERIALS AND METHODS

In this section we explain the materials and methods used for our NGS data analysis used by PMR and other Hadoop based applications.

2.1 NGS Data Sets

We used paired-end RNA-Seq data from the Flemington lab, from a Human Burkitt's lymphoma cell line [23]. The read length is 100 for both directions of paired-end data, and the total size of each data set is about 20.1 GB containing 70,927,422 reads. For varying size of sequencing data experiments, we chunked the required amount of reads by partitioning the entire data. For example, 2 GB paired end data implies 1 GB for one end and 1 GB for the other end. The file format of sequencing data is fastq and all manipulation of sequencing data was conducted by in-house python scripts and other tools such as samtools [24]. The reference genome is indexed based on the alignment tool used (BWA and Bowtie) and pre-staged to the machines before read mapping is performed.

2.2 HPC Resources

For this work, we used two Futuregrid systems – Sierra and Hotel. More details on the specification of these machines can be found from the web page of Futuregrid [25]. In brief, two systems are multi-node clusters and equipped with the PBS job scheduler. For each cluster, Hadoop is also installed. Overall, the systems are appropriate to test our PMR (single or multiple-clusters scenarios) and to compare our PMR implementations with Hadoop-based tools. Hadoop version 0.20.2 was used with the two FutureGrid machines. A replication factor of 2 and default chunk size of 128 MB is used.

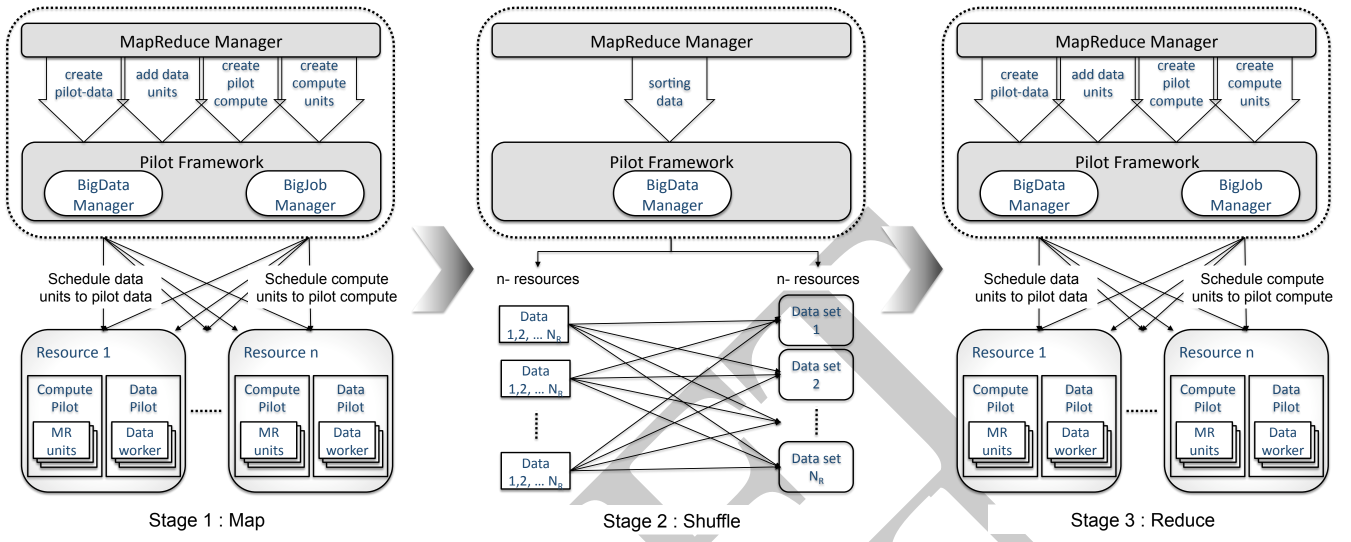


Figure 1: PMR architecture and the workflow for a MapReduce task: The compute and data units are basic blocks of scheduling in Pilot abstractions [19]. In our NGS analysis, the compute units in map phase are the alignment tasks, data units in the shuffle phase refer intermediate data units generated and compute units in reduce phase could be either duplicate read removal or SNP finding tasks.

2.3 File Systems

In our experiments, both PMR and Hadoop are executed on a shared file system. A shared file system is a common file system accessible for all the compute nodes on a single cluster. Hadoop performs well when compute node local disks are used to build HDFS, but due to limited local disk space of compute nodes we utilized shared file system to build HDFS. PMR utilize the shared file system to read/write the data, whereas Hadoop uses HDFS to read/write the data, which is built on the same shared file system.

2.4 Pilot-MapReduce (PMR)

Pilot-MapReduce is a pilot-based implementation of the MapReduce programming model, which decouples the logic/pattern of MapReduce from the actual management of the compute, data and network resources. By decoupling job scheduling and monitoring from the resource management, PMR can efficiently reuse the resource management and late-binding capabilities of BigJob and BigData.

PMR exposes an easy-to-use interface which provides the complete functionality needed by any MapReduce-based application, while hiding the more complex functionality, such as chunking of the input, sorting the intermediate results, managing and coordinating the map and reduce tasks, etc., these are generically implemented by the framework [18].

PMR supports different MapReduce topologies (i) *local* (ii) *distributed* (iii) *hierarchical*. Local PMR performs all map and reduce tasks on a single cluster. On the contrary, distributed and hierarchical PMRs are mainly intended to support distributed data processing on multiple clusters without a need of global file system.

In distributed PMR, map tasks are conducted on multiple clusters or heterogeneous infrastructures that are close to access initial input data efficiently and intermediate data are, if necessary, transferred to another resource for executing reduce tasks. Meantime, in hierarchical PMR, the output of the first phase of MapReduce are combined using the final MapReduce step. We reported in Ref. [18] the comparison between hierarchical PMR and distributed PMR in Ref. with respect to the type of workload or data aggregation involved. In this work, we mainly focus on local and distributed PMR since most of complex analyses associated with NGS data are not obvious to benefit from hierarchical MR. In Fig. 1, we illustrate the typical MapReduce workflow with PMR that can also summarized as follow.

- PMR launches map tasks on N clusters using the Pilot-Job of Pilot framework.
- After the map stage, PMR shuffles intermediate data between clusters using Pilot-Data of Pilot framework
- Finally,PMR launches reduce tasks on a cluster using the Pilot-Job of Pilot framework.

2.5 Target NGS Analysis

In this section we present the important steps of DNA sequencing that we consider for our work.

2.5.1 *Short Read Alignment*

Short reads alignment and the de-novo assembly are the required first steps in every pipeline software tool that aims to analyze sequencing data from NGS platforms. De-novo assembly still remains a challenge, because of complications arising from the short length of sequencing reads from NGS machines. In most of situations, read alignment (or mapping process) is the first task of NGS workflows, and two Hadoop-based tools, Seqal and Crossbow provided two mapping tools, BWA and Bowtie, respectively.

In general, for RNA-Seq data analysis, in particular with eukaryote genomes, the spliced aligner such as TopHat [26] is used. In our work, we consider an alternative strategy, to use a non-spliced aligner and later splicing events are detected separately, justifying the use of non-spliced aligners such as BWA and Bowtie for the RNA-Seq data. These non-spliced aligner tools mapped reads onto human reference genome hg19.

	PMR[18]	**Seqal**[10]	**Crossbow**[9]	**CloudBurst**[7]	**GATK**[8]
Key Parallelism Strategy	Pilot-based Distributed MR	Hadoop-based/ MR (Pydoop)	Hadoop Streaming	Hadoop-based MR	MR-based Structured Programming Framework
Hadoop Requirement	No	Yes	Yes[1]	Yes	No
Multiple Cluster Support	Yes	Limited by Hadoop	Limited by Hadoop	Limited by Hadoop	Limited by JVM
Multiple Node Task Supprt	Support	Not allowed by Hadoop	Not allowed by Hadoop	Not allowed by Hadoop	Not Easy
Distributed Job and Data Coordination	Advert Service (SAGA)	Hadoop/HDFS	Hadoop/HDFS	Hadoop/HDFS	Java Framework
Primary Aligner	BWA, Bowtie, and Others (coming)	BWA	Bowtie	RMAP	BWA
Multiple Aligner Support	Straightforward	Not Straight-forward	Possible	Not Straight-forward	Straight-forward
Primary Tasks	Alignment/Duplicate Removal (and Extensible for RNA-Seq)	Alignment/ Duplicate Removal	Alignment/ SNP Discovery	Alignment	Various NGS Data & Downstream Analysis
Extensibility for Supporting Multiple Tools (e.g. Multiple Aligners)	High	Medium	Low	Low	High

Table 1: Feature comparison of PMR with other tools for NGS data analysis that primarily provide a parallelism support using the MapReduce framework. [1]The feature of Crossbow that can run with a desktop environment without Hadoop is not considered in the scope of this work due to the lack of scalability support.

2.5.2 *Post-Alignment*

Duplicate read removal step might be required after short read alignment, because sample preparation processes before sequencing might contain artifacts stemming from high-throughput read amplification; many duplicates introduced are not relevant to true biological conditions.

Seqal is a Hadoop MapReduce application which implements the alignment in map phase using BWA aligner and a duplicate removal step using the same criteria as the Picard MarkDuplicates [10, 22] in reduce phase.

Crossbow [9] is a scalable software automatic pipeline, combines Alignment and SNP finding tools for DNA sequencing analysis. Crossbow contains 4 steps - preprocessing, Alignment, SNP finding and postprocessing. Each step is a Hadoop streaming-based MapReduce application and the output of each step is stored in HDFS and read from HDFS by the next step. In our experiments we focused on Crossbow alignment which uses Bowtie aligner in map phase and has a dummy reducer.

2.6 Experiments: Motivation and Configuration

Our experiments compare PMR-based approach with Hadoop API based Seqal and Hadoop streaming based Crossbow (see Table 1) to:

- Compare scale-out, scale-up and scale-across (as applicable) on distributed cyberinfrastructure using various NGS analytic tools
- Investigate extensibility to support multiple NGS analytic tools
- Compare PMR support for fine and coarse grained parallelism.

Seqal and Crossbow are different from other tools such as Cloudburst and GATK (as summarized in Table 1), in terms of usage and the type of parallelism they employ for NGS data processing and management.

For our experiments, Seqal part of Seal package of version 0.2.3 version and Crossbow package of version 1.1.2 is used. Crossbow 1.1.2 uses Bowtie 0.12.7v and SOAPsnp 1.02v tools. Default configuration or execution options are used for both the packages.

Preprocessing is a common step for both Seqal and Crossbow. The input file format for Seqal is prq format. We use the PairReadsQSeq tool of the Seal package to convert qseq/fastq format files to prq format. Crossbow also performs preprocessing on input fastq files and then performs alignment and SNP finding.

We developed a PMR based pipeline which implements the similar capabilities of Seqal and Crossbow. The PMR based pipeline for Seqal capabilities has a custom script for the map phase which performs short read alignment using BWA aligner and a script for the reduce phase to perform duplicate read removals [22]. Note that our reduce phase implementation differs from the Seqal implementation's since ours are in-house scripts. The PMR based pipeline aiming to provide the similar capabilities of Crossbow has an alignment step supporting Bowtie aligner and implements SNP finding with samtools. Note that Crossbow uses SOAPsnp along with Bowtie alignment. SOAPsnp accepts the output format of SOAPaligner as a default, but Crossbow modified Bowtie for generating outputs that can be accepted by SOAPsnp. Because of that, we do not compare SNP finding step in this work.

When executing Hadoop based MapReduce NGS tools on production cyberinfrastructure, we request a set of nodes to start the Hadoop cluster. Once required resources are obtained, we set the number of map and reduce tasks(N_W) per node and number of nodes via Hadoop configuration or NGS analytic tool configuration. Once the Hadoop cluster is ready, NGS tools can be executed. Hadoop MapReduce environment setup involves significant amount of efforts and it is not trivial for a new user.

PMR provides a simple framework which can be easily deployed on multiple clusters. PMR inherits the advantage of Pilot abstractions which liberates the application/user from the resource management. Some of the parameters involved in PMR setup are spec-

Experiment motivation (figure)	**Experiment(# of Nodes : Total # of Cores , # of Workers , # of Mappers) [case] ==> E(N_{node} : N_{core} , N_W , N_M) [case]**
Scale-out (Fig 2)	E(16:128 , 32 , 32) [10GB read size] E(16:128 , 32 , 64) [20GB read size] E(16:128 , 32 , 128) [40GB read size]
Scale-up (Fig 3)	E(4:32 , 8 , 32) [4 nodes] E(8:64 , 16 , 32) [8nodes] E(16:128 , 32 , 32) [16 nodes]
Scale-across (Fig 4)	E(4:32 , 8 , 13) [2GB read size] E(4:32 , 8 , 25) [4GB read size] E(4:32 , 8 , 50) [8GB read size]
Extensibility (Fig 5)	E(4:32 , 8 , 13) [2GB read size] E(4:32 , 8 , 25) [4GB read size] E(4:32 , 8 , 50) [8GB read size]
Parallelism (Fig 6)	E(2:16 , 4 , 50) [2 nodes] E(4:32 , 8 , 50) [4 nodes] E(8:64 , 16 , 50) [8 nodes]

Table 2: Description of experimental cases examined in this paper for understanding scalability, impact of parallelism strategy, and performance comparison. A experiment will be described with the three parameters in E(N_{node} : N_{core} , N_W , N_M). The related experiments in figures are illustrated in this table. Note that all nodes for this study have 8 cores, 8 reducers and the number of cores per Worker is determined by the total number of cores divided by the number of Workers in a node.

ifying the map and reduce scripts and their arguments, number of cores for each map and reduce task, total number of compute nodes for entire MapReduce application and the maximum estimated time (wall time) required to run the MapReduce, nature of the map or reduce task (single or mpi).

There is no limitation on the number of cores that can be assigned to a map/reduce task, where as Hadoop places a hard-limit on the number of cores on a map task as the number of cores per node. This affects the performance of the MPI based NGS tools which scale-up well. For example NovoalignMPI [27] is a message passing version of Novoalign, with claims of being one of the most accurate aligners, allows a single alignment process to spread across multiple compute nodes.

The default input split/chunk size used by Hadoop based applications is 128MB. In the PMR based applications, we make sure the right amount of chunk size is specified so that the number of map tasks are equal in both Hadoop and PMR approaches. The chunk size for PMR is considered as number of reads. The number of mappers(N_M) depend on the size of input and chunk size. The number of reduces is set to 8 in our experiments. For the relation between parameters used for the experimental set-up, see examples in Table2.

Our PMR-based pipelines directly use the fastq format files and no preprocessing required. The intermediate data management affects the performance of NGS tools in distributed data processing. PMR uses Pilot-Data which transfer the intermediate data in parallel and across multiple clusters. Hadoop depends on HDFS and TCP/IP based communication between data nodes.

3. EXPERIMENTS AND EVALUATION

We evaluate the performance of the PMR-based applications in the context of scalability at different levels and extensibility to multiple tools. Each experiment is repeated at least three times.

3.1 Scalability

The total runtime for a PMR task comprises the steps to chunk input data, map, shuffle and reduce phase times. The scalability characteristics of PMR, for different levels of parallelism and granularity were evaluated using applications specified in Section 2.

Fig 2, not surprisingly, shows the increment of required total run-

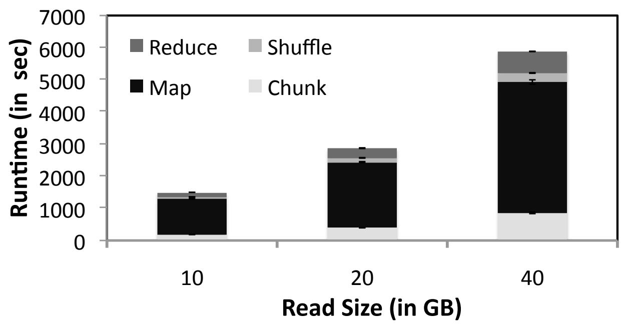


Figure 2: Need of a scalable solution with increment of data volume. Read size dependent time-to-solution is measured for MapReduce tasks of read alignment and duplicate read removal. The number of nodes for this experiment,N_{node} is 16(128 cores), the number of Workers, N_W is 32, and the chunk size is set to contain 625000 reads. The number of Reducers is set to 8. BWA is used for read alignment.

time as the input read size increase, indicating the need of a scalable solution such as MapReduce . With this experiment, we also found that PMR *scales-out* well as the number of compute tasks is increased over 16 nodes(128 cores). Meanwhile, Fig 3 shows how PMR *scales-up*. PMR scales-up well as the runtime decreases with increase in number of nodes.

In both experiments, the map phase (read alignment) appears to be a single dominant step for the overall time-to-solution, whereas the reduce task (duplicate read removal), along with other tasks, is a relatively less compute intensive task. The intermediate data is managed by Pilot-Data which parallelizes the intermediate data transfers between map and reduce phases within a cluster or between clusters..

In Fig 4, we compare results using different configurations of PMR along with results using Seqal. The setup phase for Seqal involves tasks for copying of the reference information to all the data nodes and extracting them to allow the reference available locally, whereas the setup phase for PMR involves chunking of data.

Notably, Seqal takes more time than local or distributed PMRs. This is, in our FutureGrid environment, because we are forced to have HDFS with the shared file system. The local disk of compute nodes available on FutureGrid is too small for dealing with input

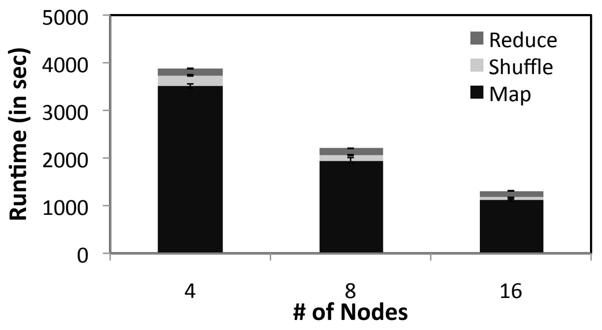


Figure 3: PMR scalability of a MapReduce-based computation for read alignment and duplicate read removal. The number of workers N_W per node is set to 2 and the number of cores per each node is 8. The input read file is 10GB, the number of Reducers is set to 8, The number of reads in a chunk is 625000. BWA is used for read alignment

data and other produced data sets, resulting in that HDFS had to be configured to use a shared file system and therefore a non-optimal performance during the I/O intensive map phase.

Hadoop is a prominent framework for carrying out MapReduce computations but it is designed primarily for a single cluster/local environment, but not straightforward for its deployment with a high degree of distributed heterogeneous resources. For example, Hadoop cannot be executed on more than one cluster on Futuregrid, because of firewall issues.

The distributed PMR, on the other hand, supports the capability of running MapReduce programming model on multiple clusters without a global filesystem between the clusters. In our experiments, distributed PMR is executed on Sierra and Hotel, where the number of workers and the input data is distributed equally between the machines.

Distributed PMR performs less than local PMR; this is apparently due to the overhead of distributing tasks between two clusters. The other reason is that the time taken to transfer intermediate data from Hotel to Sierra by Pilot-Data, which is produced by the map phase on Hotel.

Nonetheless, the observed overhead is not significant, and considering the advantages of *scaling-across* that allows the concurrent usage of multiple cluster resources, distributed PMR could be a valuable option for achieving the scalable solution.

In the case of distributed PMR, once entire intermediate data are produced, the reduce phase is executed on Sierra. Note that a direct comparison of the reduce phase is not obvious due to the different implementations between Seqal and our custom scripts for duplicate read removal tasks , in spite of the fact that the same criteria of duplicate read removal of Seqal[22] was used for our implementation. The map phase of Seqal and PMR are almost comparable since the same aligner, BWA is used and all parameters (see Table 2) for configuring a MapReduce task are set to be same for the comparison.

3.2 Extensibility

The extensibility of PMR is demonstrated with two aligners – BWA and Bowtie. One of the important reasons why multiple aligners are needed is because of the difficulty of validation of an aligner used[28]. It is well studied that each aligner implements different strategies to deal with the requirement of computational loads, memory usage, and sensitivity associated with decision on algorithms and computational implementations of indexing, search, and match tasks.

Indeed, the decision of which aligner affects not only alignment results but also investigate downstream analysis that aim to study genome variation, transcriptome analysis, and DAN-protein interactions. Therefore, it is not an overstatement to emphasize the importance of supporting multiple tools as well as providing an effective means for implementing such tools within a reasonably short development period for infrastructure of NGS data.

Fig. 5, evaluates the performance of read alignment in the map phase of both Hadoop and PMR based applications for Bowtie and BWA aligners. Hadoop implementations - Crossbow uses Bowtie aligner and Seqal uses BWA aligner. Custom python wrappers to Bowtie and BWA aligner are developed to execute alignment in the map phase of PMR. In the evaluation, both Hadoop based implementations face the problem of non-optimal configuration of Hadoop, i.e usage of shared file system for HDFS, where as both local and distributed PMR perform better than Hadoop map phase for both aligners. The PMR is extensible and can support multiple NGS analytic tools.

Extending PMR to support new NGS analytic tools involve development of simple map and reduce wrapper scripts to the tools. The wrapper scripts could be developed in any language. To some extent, Hadoop streaming supports this types of extensibility but still requires complexity of managing computational resources to maintain Hadoop cluster. PMR liberates the user from the complex task of maintaining and acquiring computational resources and executing map and reduce tasks on them.

3.3 Parallelism

PMR supports multiple levels of parallelisms – thread, task and multiple-cores, and enables the flexible configurations of codes. For example, BWA and Bowtie can be invoked to use varying number of threads (fine-grained parallelism). In Fig. 6, we showed how such options could affect the performance. Even though it is feasible for other tools such as Seqal or Crossbow to handle such options, the PMR approach of separating the runtime environment (Pilot) from the code invocation in the map and reduce phases, provides the capability of utilizing the fine-grained parallelism along with the coarse grain parallelism provided by MapReduce. The fine grain parallelism provided by Pilot-Job framework is demonstrated in replica exchange implementation [29].

One of the advantages of PMR is it doesn't impose any restriction on number of compute nodes that can be assigned to a particular map or reduce task. This leads to a natural and native support for MPI-based NGS tools. For example, NovoalignMPI [27] is a message passing version of Novoalign, with claims of a more accurate aligner, allows a single alignment process to use multiple compute nodes. The MPI versions of Novoalign are more beneficial when large computing infrastructures are available. Hadoop doesn't provide flexibility to assign multiple compute nodes to a single compute task, thus leading to an impedance mismatch between Hadoop MR and MPI based NGS analytic tools.

Related Work

Cloudburst was one of the first generation tools in this field and demonstrated the significant potential of the MapReduce model for NGS data analysis. After Cloudburst, Crossbow was developed by focusing on better scalability using Hadoop streaming. Compared to the two Hadoop-based tools, GATK was introduced to address the general parallel execution pattern across many NGS data analytics. Recently, the Seal package was announced in which Seqal is a utility for the alignment and duplicate read removal.

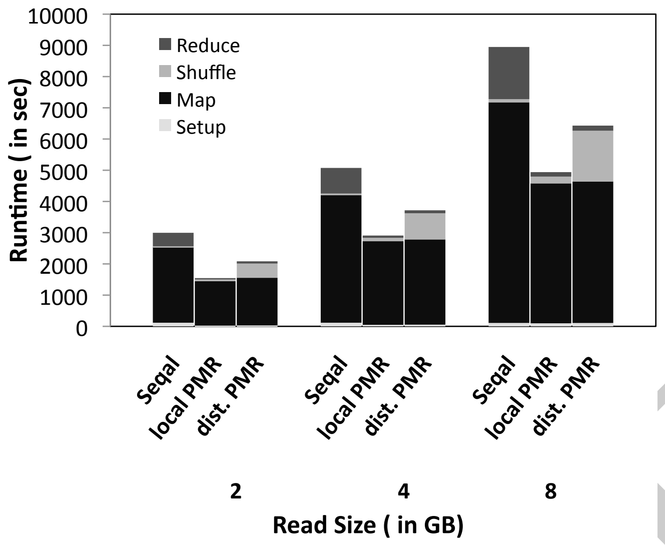
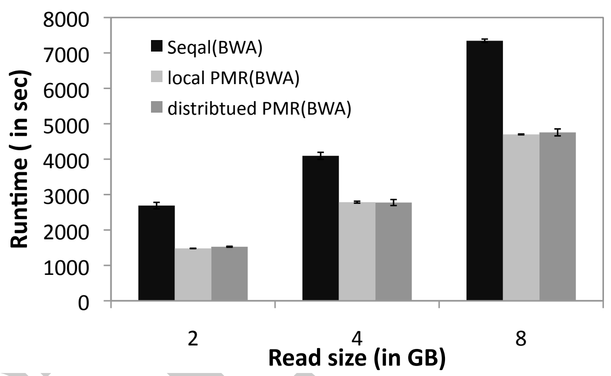


Figure 4: Comparison of the total time-to-solution for a MapReduce-based computation of alignment and duplicate read removal. Hadoop-based Seqal is compared to local-PMR vs. distributed-PMR. For this experiment, the number of Nodes N_{node} is 4, the number of Workers N_W is 8, the number of Reducers is 8, and the number of reads in each chunk is 292,763. For the distributed-PMR, two machines of FutureGrid, Sierra and Hotel were used, whereas Sierra was used for other cases.



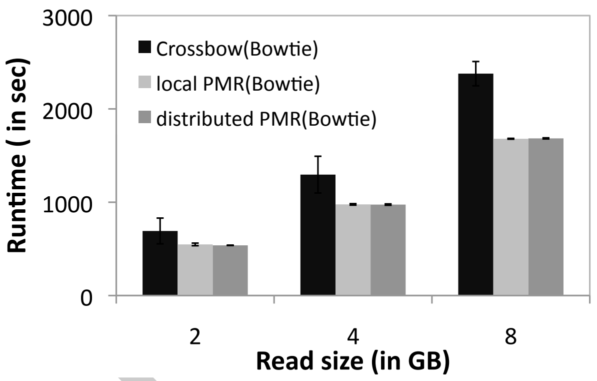


Figure 5: Comparison of runtimes for the map phase. The map phase of Seqal, local-PMR(BWA), distributed-PMR(BWA), local-PMR(Bowtie), distributed-PMR(Bowtie), and Crossbow(Bowtie) are compared. The aligner used for each case is indicated in a parenthesis. For this experiment, the number of nodes, N_{node} is 4, the number of Workers, N_W is 8, and the number of reads in each chunk is 292,763. For the distributed-PMR, two machines of FutureGrid, Sierra and Hotel were used, whereas Sierra was used for other cases.

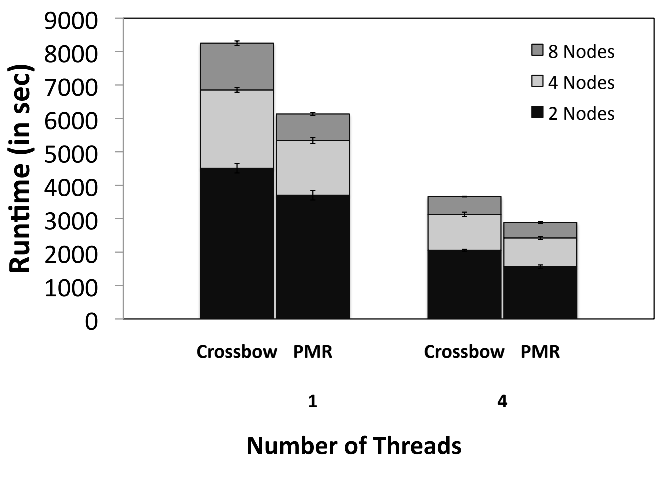


Figure 6: The map phase runtimes of PMR(Bowtie) and Crossbow(Bowtie) are compared, by varying number of threads for each map task. Number of workers/Node = 2 and input data size is 8 GB. The maximum number of cores assigned to a worker is 4, so we used 4 threads to achieve maximum fine-grain parallelism

4. DISCUSSION

DNA sequencing data generated by NGS platforms[1–5]. has resulted in a fundamental change in the amount of data that needs to be analyzed. Such high-throughput technologies are able to provide information about an entire genome or statistically meaningful number of genomes of same species or related species.

While there have been algorithmic advances and a plethora of new software tools by user-friendly interface via web-based tools, GUI tools, or computing environments [15], the end-to-end development of scalable infrastructure is lagging. MapReduce is a widely employed programming model for parallelization of a large data process; as reported by others with the tools listed in Table 1, we observed that an efficient runtime environment for MapReduce such as that provided by PMR enables efficient methods for dealing with the required scalability of NGS data analysis which comprises short reads alignment and other following analysis such as duplicate read removal and SNP finding.

PMR – A viable solution for scale-across and extensible NGS analytics: In fact, PMR not only supports scaling-across, it provides some unique features, viz., support for distributed data analysis and multiple tools that can each exploit multiple levels of parallelism. PMR provides an extensible runtime environment with which minimally modified, yet standalone target tools are executed and the overall workflow can be dynamically optimized by exploiting multiple levels of parallelism. Furthermore, as indicated by results for BWA and Bowtie for alignment, PMR allows further extensions of existing implementation with other complementary tools or a flexible pipeline development. In fact, our primary goal behind this work is to develop an integrative pipeline service for RNA-Seq data, and our development presented in this work is indicative of preliminary progresses toward such a goal.

PMR vs. Hadoop-based MapReduce tools: The open source Hadoop provides an ideal environment for the tools based on the

MapReduce programming model. Perhaps, the ease of installation with commodity hardware and the robust stability with fault-resiliency and easy scalability could propel a wide acceptance of Hadoop. Nonetheless, Hadoop-based approaches find many hurdles in distributed contexts and scenarios: for example, when scaling across multiple clusters[30], the execution of applications across multi-nodes such as one using MPI, and the solutions involving the limitations imposed by a current Hadoop implementation need to be addressed.

As evidenced by the comparison of PMR-based read alignment and a following analysis with two Hadoop-based tools – Crossbow and SEAQL, PMR-based implementations demonstrate scalability advantages over Hadoop-based approaches. It is likely that future releases of Hadoop address the distributed data scenarios too.

DARE and beyond: PMR-based NGS tools, implemented and scrutinized in this work, were developed in conjunction with our development of the runtime environment for dynamic applications, Distributed Application Runtime Environment (DARE) [31, 32]. DARE is a strategically important component for the development of Science Gateway for NGS data analytics and downstream analysis. Under the design strategy of DARE-based gateways, PMR-based tools were conceived to be a main category of supporting execution patterns for parallel and distributed task and data management.

Acknowledgement

The project described was partially supported by Grant Numbers HPCOPS NSF-OCI 0710874 award, NSF-ExTENCI (OCI-1007115) P20RR016456 from the NIH National Center For Research Resources. Important funding for SAGA has been provided by the UK EPSRC grant number GR/D0766171/1 (via OMII-UK) and the Cybertools project (PI Jha) NSF/LEQSF (2007-10)-CyberRII-01. Computing resources were made possible via NSF TRAC award TG-MCB090174 and LONI resources. This document was developed with support from the National Science Foundation (NSF) under Grant No. 0910812 to Indiana University for "FutureGrid: An Experimental, High-Performance Grid Test-bed." We also acknowledge the Seal developer, Luca Pireddu for useful performance related discussions, and Erik Flemington for allowing us to use his RNA-seq data sets.

References

[1] M. L. Metzker. Sequencing technologies - the next generation. *Nat. Rev. Genet.*, 11(1):31–46, 2010.

[2] The 1000 Genomes Project Consortium. A map of human genome variation from population-scale sequencing. *Nature*, 467:1061–1073, 2010.

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

[4] A. Bateman and J. Quackenbush. Editorial -Bioinformatics for Next Generation Sequencing. *Bioinformatics*, 25:429, 2009.

[5] J. D. McPherson. Next-Generation Gap. *Nature Methods*, 6:s2–s5, 2009.

[6] Michael C. Schatz, Ben Langmead, and L. Salzberg, Steven. Cloud computing and the DNA data race. *Nature Biotechnology*, 28:691–693, 2010.

[7] M. C. Schatz. CloudBurst: highly highly sensitive read mapping with MapReduce. *Bioinformatics*, 25(11):1363–1369, 2009.

[8] 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.

[9] B. Langmead, M. C. Schatz, J. Lin, M. Pop, and S. L. Salzberg. Searching for SNPs with cloud computing. *Genome Biol.*, 19(11):R134, 2009.

[10] L. Pireddu, Leo S., and G. Zanetti. SEAL: a distributed short read mapping and duplicate removal tool. *Bioinformatics*, 27(15):2159–2160, 2011.

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

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

[13] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. In *Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation - Volume 6*, pages 10–10, Berkeley, CA, USA, 2004. USENIX Association.

[14] Andrew Thrasher, Rory Carmichael, Peter Bui, Li Yu, Douglas Thain, and Scott Emrich. Taming complex bioinformatics workflows with weaver, makeflow, and starch. *Workshop on Workflows in Support of Large Scale Science*, pages 1–6, 2010.

[15] J. Goecks, A. Nekrutenko, J. Taylor, and et al. A Comprehensive Approach for Supporting Accessible, Reproducible, and Transparent Computational Research in the Life Sciences. *Genome Biol.*, 11(8):R86, 2010.

[16] http://www.saga-project.org/.

[17] Saurabh Sehgal, Miklos Erdelyi, Andre Merzky, and Shantenu Jha. Understanding application-level interoperability: Scaling-out mapreduce over high-performance grids and clouds. *Future Generation Computer Systems*, 27(5):590–599, 2011.

[18] Pradeep Mantha, Andre Luckow, and Shantenu Jha. Pilot-MapReduce: An Extensible and Flexible MapReduce Implementation for Distributed Data, 2012. Accepted to MapReduce workshop 2012 (HPDC12).

[19] Andre Luckow, Mark Santcroos, Sharath Maddineni, Andre Merzky, and Shantenu Jha. P*: An Extensible Model of Pilot-Abstractions for Dynamic Execution. In *under review*, 2011.

[20] Yuan Luo, Zhenhua Guo, Yiming Sun, Beth Plale, Judy Qiu, and Wilfred W. Li. A hierarchical framework for cross-domain MapReduce execution. In *Proceedings of the second international workshop on Emerging computational methods for the life sciences*, ECMLS '11, pages 15–22, New York, NY, USA, 2011. ACM.

[21] http://hadoop.apache.org/.

[22] Luca Pireddu, Simone Leo, and Gianluigi Zanetti. Mapreducing a genomic sequencing workflow. In *Proceedings of the second international workshop on MapReduce and its applications*, MapReduce '11, pages 67–74, New York, NY, USA, 2011. ACM.

[23] Z. Lin, G. Xu, N. Deng, C. Taylor, D. Zhu, and E. K. Flemington. Quantitative and qualitative RNA-Seq-based evaluation of Epstein-Barr virus transcription in type 1 latency Burkitt's lymphoma cells. *J. Virol.*, 24:13053–8, 2010.

[24] H. Li, B. Handsaker, A. Wysoker, T. Fennell, J. Ruan, N. Home, G. Marth, G. Abecasis, R. Durbin, and 1000 Genome Project Data Processing Subgroup. The Sequence alignment/map (SAM) format and SAMtools. *Bioinformatics*, 25:2078–2079, 2009.

[25] FutureGrid. `http://www.futuregrid.org`.

[26] S. Pepke, B. Wold, and A. Mortazavi. Computation for ChIP-seq and RNA-seq studies. *Nature Methods*, 6:S22–S32, 2009.

[27] http://www.novocraft.com/.

[28] H. Li and N. Homer. A survey of sequence alignment algorithms for next-generation sequencing. *Briefings in Bioinformatics*, 11(5):473–483, 2010.

[29] A. Luckow, S. Jha, J. Kim, A. Merzky, and B. Schnor. Adaptive Replica-Exchange Simulations. *Royal Society Philosophical Transactions A*, pages 2595–2606, June 2009.

[30] Michael Cardosa, Chenyu Wang, Anshuman Nangia, Abhishek Chandra, and Jon Weissman. Exploring MapReduce efficiency with highly-distributed data. In *Proceedings of the second international workshop on MapReduce and its applications*, MapReduce '11, pages 27–34, New York, NY, USA, 2011. ACM.

[31] Joohyun Kim, Sharath Maddineni, and Shantenu Jha. Building Gateways for Life-Science Applications using the Dynamic Application Runtime Environment (DARE) Framework. In *Proceedings of the 2011 TeraGrid Conference: Extreme Digital Discovery*, TG '11, pages 38:1–38:8, New York, NY, USA, 2011. ACM.

[32] 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.