# Scaling with the flow

## Advantages of a MapReduce-based sequencing workflow

L. Pireddu*[1], S. Leo[1], F. Reinier[1], R. Berutti[1,2], R. Atzeni[1], and G. Zanetti[1]
[1]CRS4, Polaris, Ed. 1, I-09010 Pula, Italy
[2]Dept. of Biomedical Sciences, University of Sassari, Italy.
*luca.pireddu@crs4.it

A.D. MDLXII

## Introduction

The CRS4 Sequencing and Genotyping Platform (CSGP) has grown relatively quickly to its present sequencing capacity with 6 Illumina sequencers. This growth revealed weaknesses in the scalability of its processing workflow, which we think is similar to the workflows used in many other sequencing centers around the world.

Given the new and growing requirements in processing throughput, the CSGP has been redisegning its workflow for scalability, adopting the Hadoop MapReduce framework, which has proven itself in many other settings, and the Seal tool suite [3]. Several advantages have been observed, and the overhaul has given CSGP the opportunity to also leverage MapReduce for downstream analyses such as genotyping and to improve how it manages experiment metadata.

## MapReduce Advantages

Some of the advantages of our new MapReduce workflow.

### Scalable

Need to process faster? Simply get more machines! Buy or rent a cluster to meet demands

### No High-Perf central storage

The HDFS removes the requirement for a high-performance shared storage appliance. In addition, Hadoop minimizes the need for remote disk I/O.

### High I/O throughput

With HDFS one sums the throughput of all the disks in the cluster.

### Resistant

Being based on Hadoop, the MapReduce can resist node failures and transient cluster conditions like load peaks or network outages.

## CSGP

**CRS4 Sequencing and Genotyping Platform**
The CSGP is the largest sequencing center in Italy and one of the largest in Europe. At the moment it is mainly sequencing DNA and RNA in support of population-wide studies on autoimmune diseases.

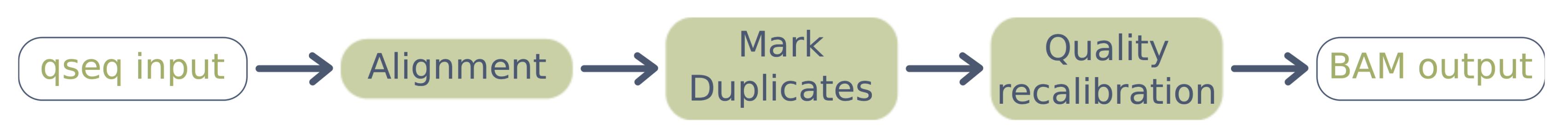
**Equipment:** 4 HiSeq2000 and 2 GAIIx Illuminas
**Capacity:** about 7000 Gbases/month
**Main activities:**
- DNA low-pass sequencing
- Exome sequencing
- RNA sequencing

**Data center:**
- 0.5 PB storage
- 400 shared computation nodes (8-core, 16 GB RAM, 2x250GB disks)

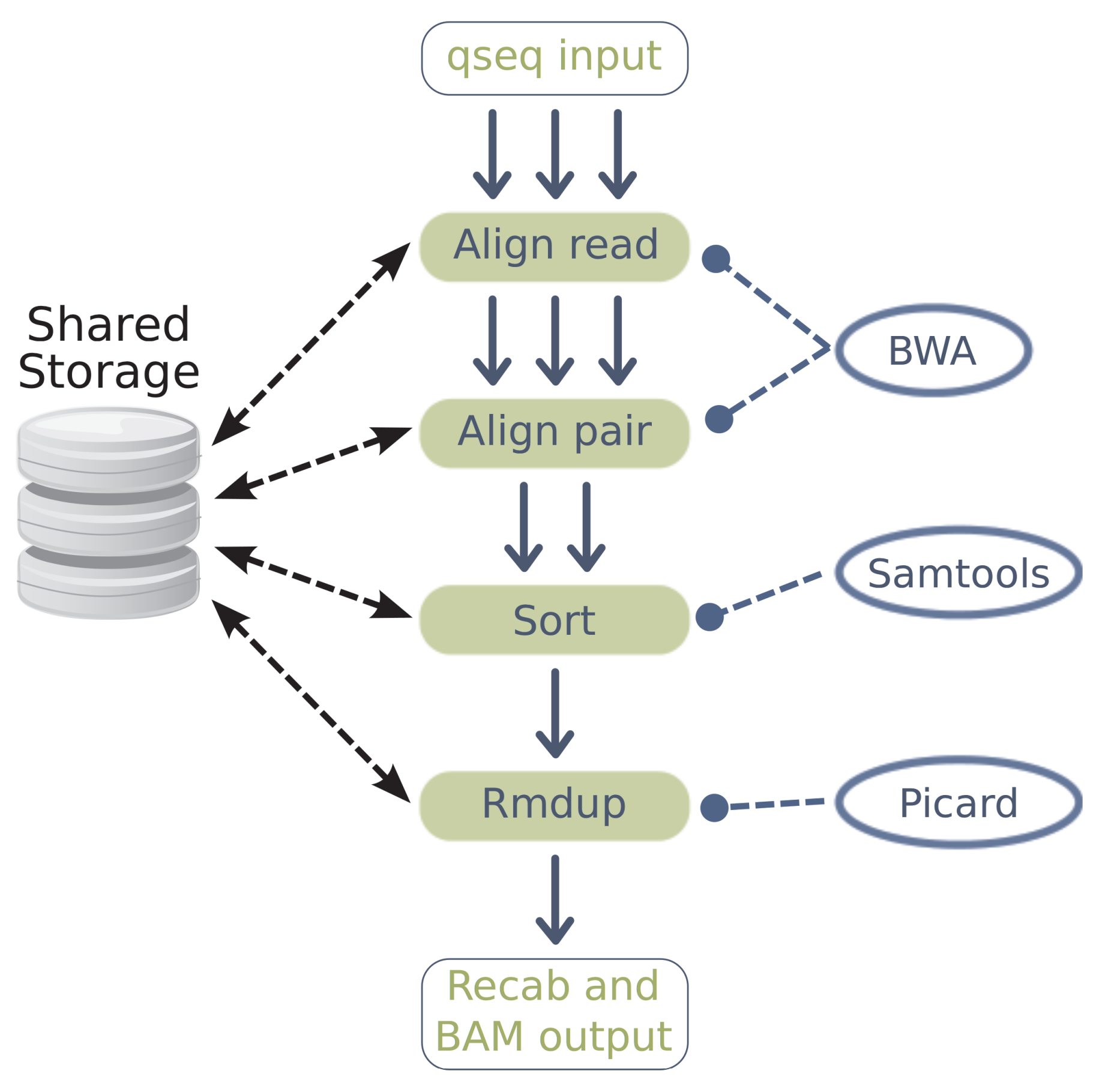
## Conceptual Workflow

Conceptually, CSGP uses the following workflow to process sequenced reads:
1) align reads to the reference;
2) identify PCR duplicates;
3) apply empirical base quality recalibration.

Other steps are "overhead" required by the implementation of the workflow. These may include:
- format conversions;
- sorting and indexing;
- copying.



## Previous workflow



In the traditional workflow each step acts like a filter, reading the output of its preceeding step and writing its own output
- Popular sequence analysis tools are used to implement each step;
- Tasks are launched via Sun Grid Engine;
- I/O is done to/from shared storage: a bottleneck that limits scalability in no. of nodes;
- I/O is not local to the computing node;
- Low parallelism: first per-lane, then per-sample;
- A failed step requires human intervention to resolve.

Note that:
- We don't show the format conversion steps (qseq to fastq, sam to bam);
- We focus on the steps that have MapReduce equivalents (Recalibration: not yet).

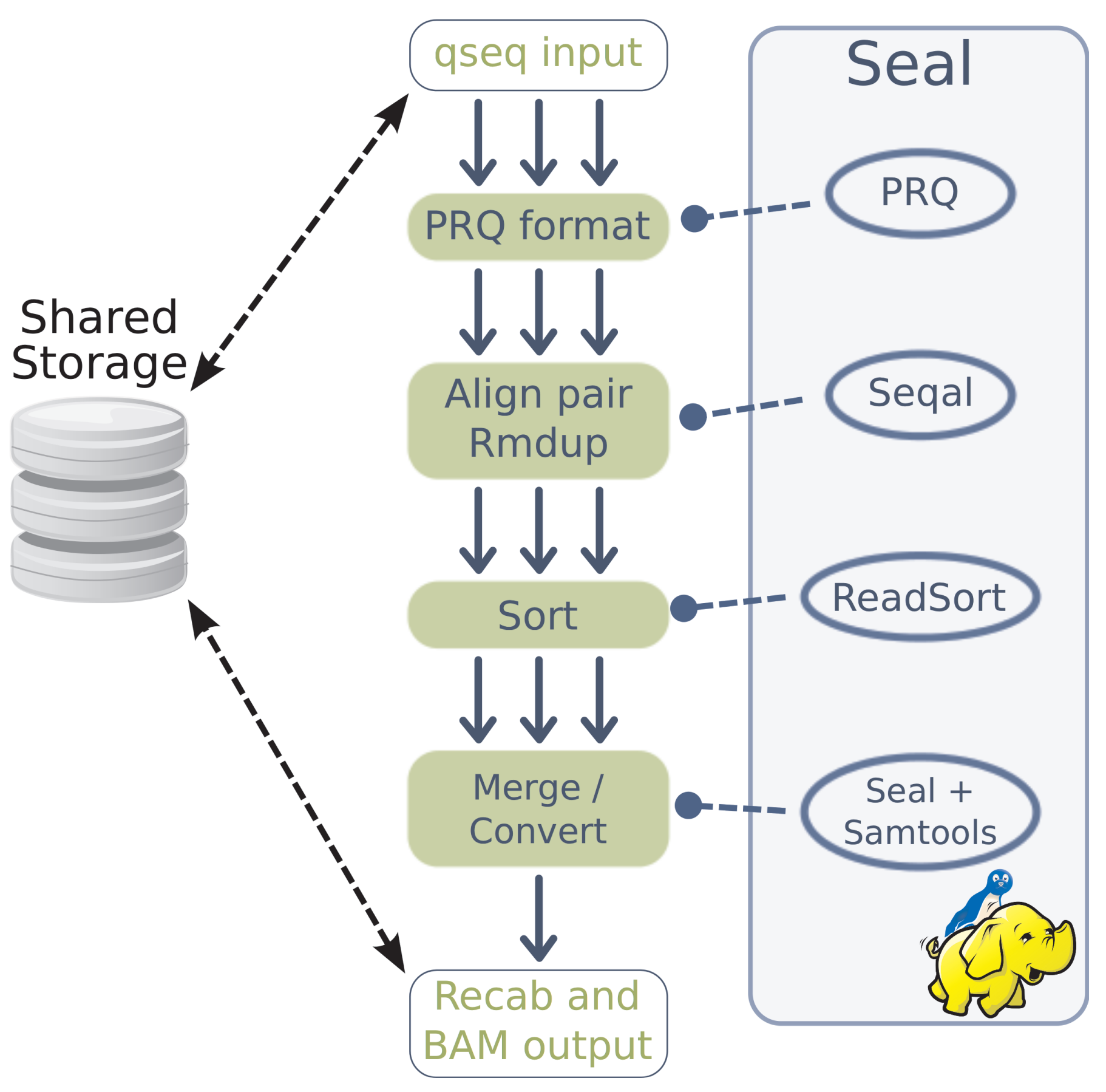
## MapReduce workflow

We have replaced whatever possible with the MapReduce-based distributed tools from Seal [3] and HDFS.

| Original | Seal replacement |
|---|---|
| BWA | Seal seqal |
| Picard MarkDuplicates | Seal seqal |
| samtools sort | Seal read_sort |
| samtools merge | Seal merge_alignments |

In addition we:
- convert qseq to prq on Hadoop (put read mates in same record);
- download data from HDFS with Seal merge_alignments.

Among its advantages the new workflow includes:
- parallelism at each read pair for the alignment, and then at each read;
- robustness to node and network problems—Hadoop re-tries failed tasks automatically;
- higher throughput from improved use of computing resources.



Merging sorted data into one file is a consequence of the fact that we have yet to finish converting the entire pipeline.

## What did I do to my data?

We must document the entire path of a data set through our infrastructure: from acquisition to analysis. As a first step to achieve this goal we are working on Galaxy integration, which will allow us to easily define the workflows in a way amenable to processing. The second step is the development of our Computational Biobank (see poster [943W]) which lets us document the entire flow graph of data that enters and is processed in our center.

## Hadoop ROI

Once our Hadoop cluster was in place, we found it advantageous to shift some of our other analyses to this platform.
**Genotype calling:** we implemented a MapReduce program to replace birdseed from the Affimetrix Power Tools) reducing analysis times from 3 weeks to 12 hours in our specific case (see poster [935W]).
**Scripting for big data:** using Pydoop [2], it is relatively easy to write simple MapReduce Python scripts that automatically leverage all the cluster's computational resources.

**Sample Pydoop script**
```
from pydoop.pipes import runTask, Factory, Mapper, Reducer
# add RG tag based on read id.
class pipe_tweaks_mapper(Mapper):
    def map(self, ctx):
        line = ctx.getInputValue()
        read_id, rest = line.split("\t", 1)
        rg = read_id[0:11]
        ctx.emit(read_id, "%s\tRG:Z:%s" % (rest, rg))

class null_reducer(Reducer):
    pass

if __name__ == "__main__":
    runTask(Factory(pipe_tweaks_mapper, null_reducer))
```

## Scalability/Performance

The new MapReduce workflow improves the throughput per node from 700 to 1400 reads/sec/node. By increasing the number of dedicated nodes from 16 to 30 (1.9x) overall throughput was increased by a factor of 3.7x over the old workflow. The improvement is mainly due to the improved parallelism offered by the Seal tools and increased I/O bandwidth of HDFS.

Seal has been verified to scale well in input size and cluster size, as is shown by Fig. 2. As such, we are confident we can further increase throughput by simply adding more computing nodes.

Node configuration: dual quad-core Intel Xeon CPUs @ 2.83 GHz, 16 GB RAM, two 250 GB SATA disks (one for HDFS), and GbE.

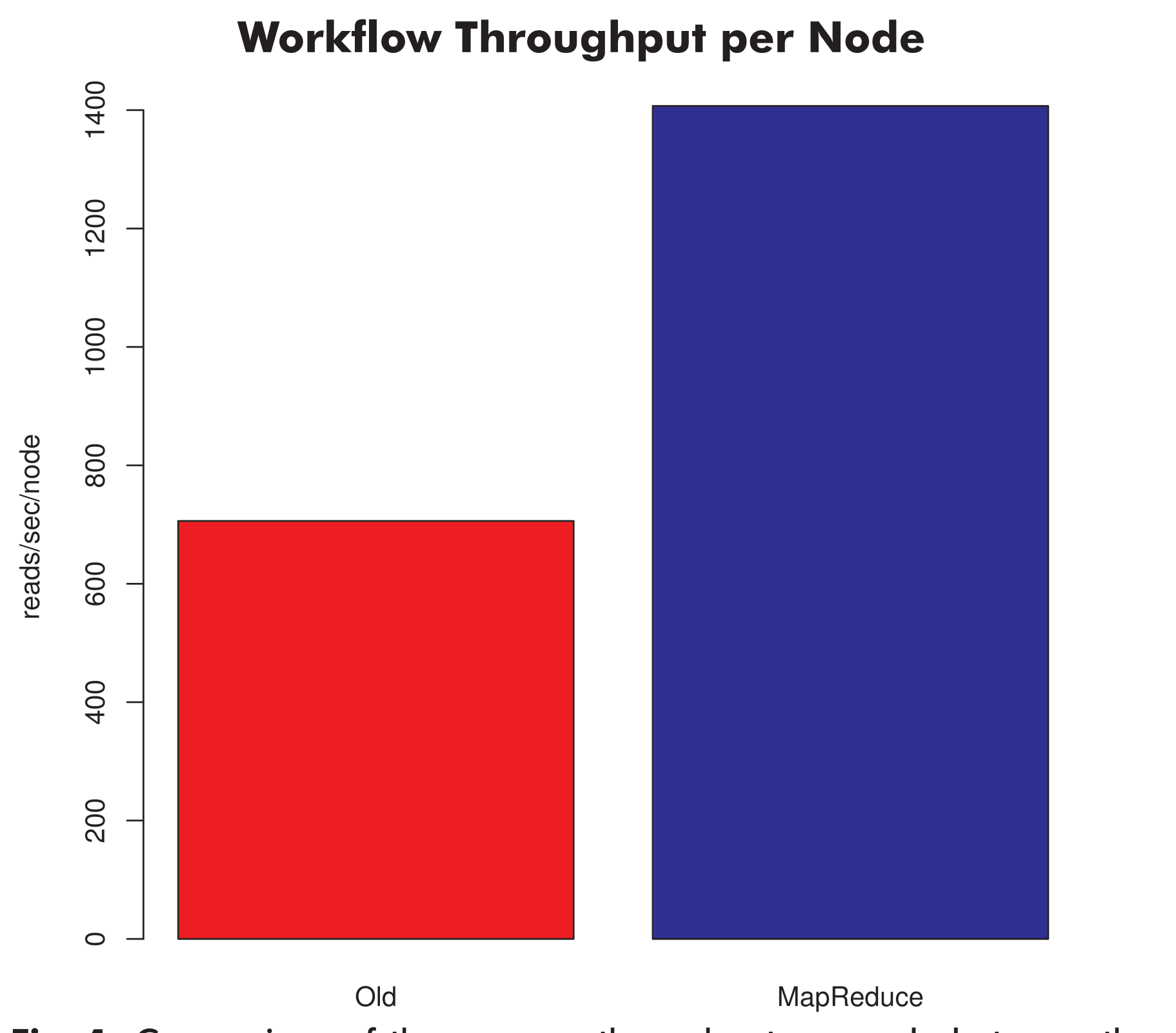


**Fig. 1.** Comparison of the average throughput per node between the old and the new workflow. The new workflow obtains such a significant improvement because thanks to its improved parallelism it is able to use all available machines for all steps implemented by Seal.

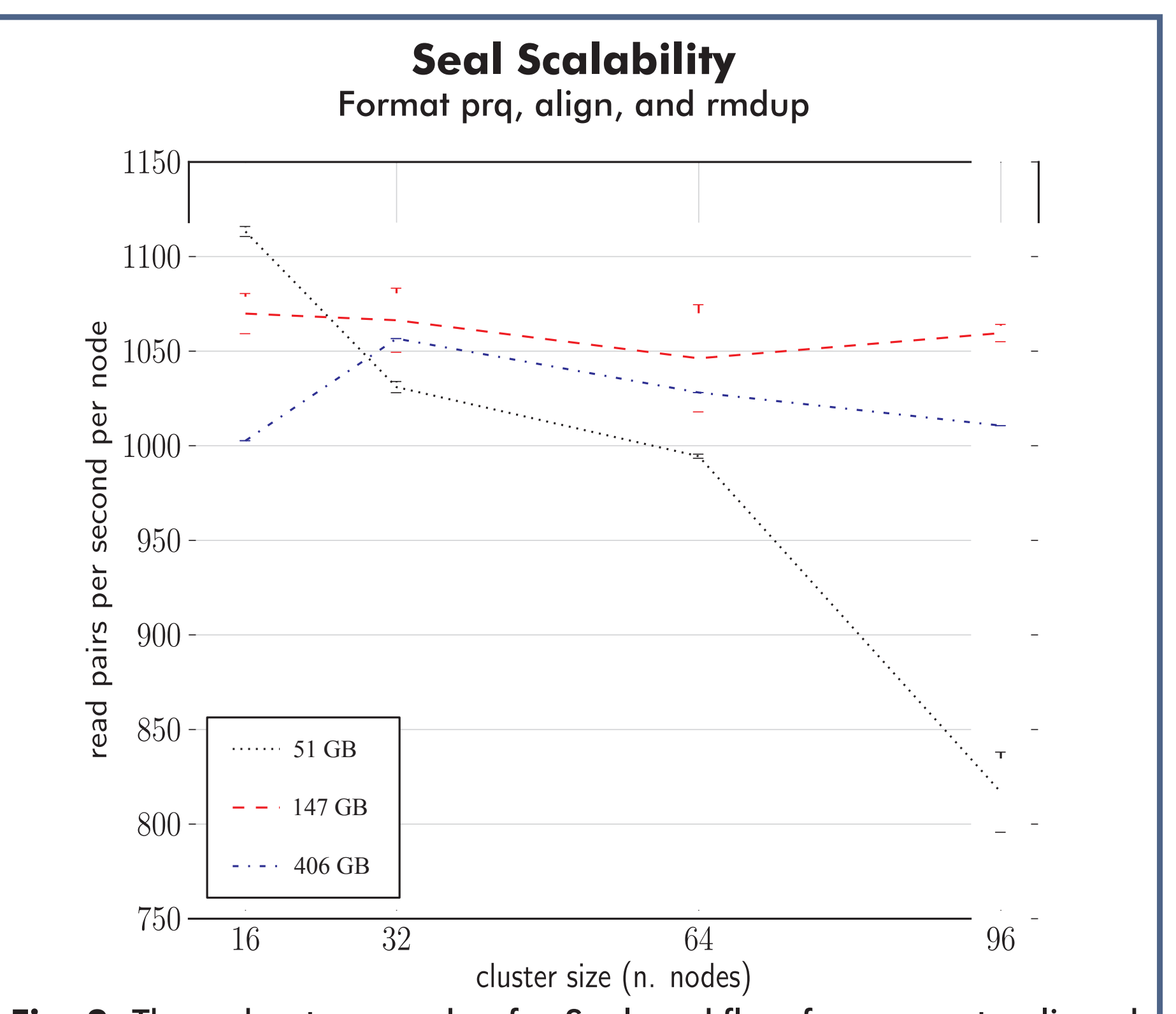


**Fig. 2.** Throughput per node of a Seal workflow from qseq to aligned SAM with marked duplicates (PRQ and Seqal applications) [3]. An ideal system would produce a flat line. By comparison, a one-node workflow using the standard multi-threaded BWA and Picard, reaches approx 1100 pairs/sec on a 5M pair data set.

## Technologies

### MapReduce and Hadoop

MapReduce is a programming model that breaks down algorithms into successions of two types of steps: map (a transformation) and reduce (an aggregation). By imposing that these functions have no side-effects and interactions, MapReduce encapsulates the functionality relative to distributing the computation into a specialized reusable framework, so that creating new scalable, distributed programs is much simplified. Hadoop is the most widespread implementation of MapReduce. In addition to providing a MapReduce implementation, it provides a distributed file system which distributes data among all cluster nodes, effectively multiplying aggregate throughput by the number of cluster nodes.

### Libbwa

Libbwa is a Python module which integrates BWA's read alignment algorithm [1]. In addition providing the same speed as BWA, multiple libbwa instances share references in memory (it can run 8 alignments in parallel with less than 16 GB of RAM using the 1KG reference) and provides a simple Python interface, making it simple to write custom read mapping scripts in Python.

### Pydoop

Pydoop [2] makes it simple to write Python MapReduce scripts that run on Hadoop and can access the Hadoop distributed file system (HDFS) directly. We use Pydoop to create fast custom data manipulation and analysis programs. In addition, Pydoop is at the base of Seal's Seqal distributed aligner.

## Conclusion

The observations presented suggest that the MapReduce programming model, Seal and Hadoop are enabling technologies that provide considerable benefits in the genomic sequencing domain:
- improved efficiency;
- improved throughput;
- reduced hardware requirements;
- improved robustness.

This last point is especially important because it brings us closer to a complete automation of the complete processing pipeline, with a significant reduction in the amount of operator time required for sequencing.
These improvements are already visible and will certainly become even more evident once the workflow is converted to MapReduce.

## References

[1] Li, H. and Durbin, R. (2009). Fast and accurate short read alignment with Burrows-Wheeler transform. Bioinformatics, 25(14), 1754–1760.
[2] Leo, S. and Zanetti, G. (2010). Pydoop: a Python MapReduce and HDFS API for Hadoop. In Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing, pages 819–825.
[3] Pireddu, L., Leo, S. and Zanetti, G. (2011). SEAL: a Distributed Short Read Mapping and Duplicate Removal Tool. Bioinformatics.
[4] Goecks, J., Nekrutenko, A., Taylor, J. and The Galaxy Team. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. Genome Biol. 2010 Aug 25;11(8):R86.
[935W] Leo, S., Zara, I., Valentini, M., Sanna, S., Zanetti, G. Scalable data management and computable framework for large scale longitudinal studies. ASHG/ICHG 2011 Meeting in Montreal, Canada, October 11-15.
[943W] Lianas, L., Cuccuru, G., Leo, S., Zara, I., Pitzalis, M., Zoledziewska, M., Deidda, F., Sanna, S., Zanetti, G. Scalable data management and computable framework for large scale longitudinal studies. ASHG/ICHG 2011 Meeting in Montreal, Canada, October 11-15.