Performance analysis of a parallel, multi-node pipeline for DNA sequencing

Journal Title
XX(X):2–10

DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Dries Decap[1,5] and Joke Reumers[2,5] and Charlotte Herzeel[3,5] and Pascal Costanza[4,5] and Jan Fostier[1,5]

Abstract

Post-sequencing DNA analysis typically consists of read mapping followed by variant calling and is very time-consuming, even on a multi-core machine. Recently, we proposed Halvade, a parallel, multi-node implementation of a DNA sequencing pipeline according to the GATK Best Practices recommendations. The MapReduce programming model is used to distribute the workload among different workers. In this paper, we study the impact of different hardware configurations on the performance of Halvade. Benchmarks indicate that especially the lack of good multithreading capabilities in the existing tools (BWA, SAMtools, Picard, GATK) cause suboptimal scaling behavior. We demonstrate that it is possible to circumvent this bottleneck by using multiprocessing on high-memory machines rather than using multithreading. Using a 15-node cluster with 360 CPU cores in total, this results in a runtime of 1h 31 min. Compared to a single-threaded runtime of ∼12 days, this corresponds to an overall parallel efficiency of 53%.

Keywords

Halvade, DNA-sequencing, Performance analysis, Parallel computing, Variant detection

Prepared using sagej.cls *[Version: 2015/06/09 v1.01]*

Introduction

Post-sequencing DNA analysis typically consists of the alignment of reads to a reference genome ('read mapping') followed by the identification of differences between the reference genome and the aligned reads ('variant calling'). For both tasks, numerous tools have been described in literature. Recently, the Broad Institute has proposed the Best Practices recommendations [12013Van der Auwera et al.] for a DNA variant calling pipeline based on BWA [22009li et al.] for read alignment, SAMtools [32009Li et al.]/Picard [42009Wysoker et al.] for data preprocessing and GATK [52010McKenna et al., 62011Depristro et al.] for variant calling. Especially for whole-genome datasets, this pipeline is very time consuming with a single-core runtime of ∼12 days to process the NA12878 dataset (Illumina Platinum genomes, 1.5 billion paired-end reads, 100 bp, 50-fold coverage, human genome). Even when enabling multithreading support in the individual tools, the execution time for this dataset is still ∼5 days on a 24-core machine (dual socket Intel Xeon E5-2695 v2 @ 2.40GHz), indicative of a poor scaling behavior.

To deal with this bottleneck, we recently proposed Halvade [72015Decap et al.], a parallel, multi-node framework in which a variant calling pipeline has been implemented according to the GATK Best Practices recommendations. Halvade relies on the MapReduce programming model [82008Dean et al.] to run multiple instances of existing tools (BWA, SAMtools/Picard, GATK) in parallel both across and within nodes on subsets of the data. Halvade is based on the simple observation that read mapping is parallel by read (i.e., aligning a certain read does not depend on the alignment of other reads) while variant calling is parallel by genomic region (i.e., variant calling in a certain genomic region does not depend on variant calling in other genomic regions). During the map phase, BWA is used to align reads to a reference genome in parallel, whereas data preprocessing (SAMtools/Picard) and variant calling (GATK) are handled during the reduce phase by operating on different genomic regions in parallel. In between the map and reduce step, the aligned reads are sorted according to genomic position using the MapReduce sorting functionality. For details about the implementation of Halvade and the tools involved we refer to [72015Decap et al.].

[1]Department of Information Technology, Ghent University - iMinds, Gaston Crommenlaan 8 bus 201, 9050 Ghent, Belgium
[2]Janssen Research & Development, a division of Janssen Pharmaceutica N.V., 2340 Beerse, Belgium
[3]Imec, Kapeldreef 75, 3001 Leuven, Belgium
[4]Intel Corporation Belgium
[5]ExaScience Life Lab, Kapeldreef 75, 3001 Leuven, Belgium

Corresponding author:
Jan Fostier, Department of Information Technology Ghent University - iMinds, Gaston Crommenlaan 8 bus 201, 9050 Ghent, Belgium
Email: dries.decap@intec.ugent.be,jan.fostier@intec.ugent.be

Prepared using sagej.cls

In [72015Decap et al.], it was demonstrated that Halvade strongly reduces the runtime: on a 15-node cluster, each node containing 24 CPU cores and 64 GB of RAM, the NA12878 is processed in 2h 39 min. Additionally, it was shown that the multi-node parallel efficiency of Halvade is excellent (around 90%), which means that the runtime is significantly reduced by using 15 nodes compared to using only a single node. However, significant performance loss can still be observed *within* each node. This can be seen from the overall performance: with a runtime of 2h 39 min using 360 CPU cores (15 nodes × 24 cores/node), a speedup of ∼108 is obtained compared to a single-threaded runtime of ∼12 days. This corresponds to an overall parallel efficiency of about 30%, suggesting the presence of certain performance bottlenecks. Understanding the performance of a sequencing pipeline is a non-trivial matter. Certain components in the pipeline are very compute-intensive (e.g. read alignment) whereas other components (e.g. data preprocessing) are mostly data-intensive. Therefore, certain tools might be CPU bound whereas others might be limited by I/O bandwidth. In order to better understand the influence of hardware configuration on the performance of sequencing pipelines, we have set up a range of benchmarks in order to identify possible bottlenecks. Specifically, in this paper, we study the influence on the total runtime of the amount of available RAM, the presence of NUMA domains, the type of network interconnection, the use of solid-state disks versus hard-disk drives and finally, the use of a distributed vs. centralized file system. We demonstrate that the use of high-memory machines and NUMA optimizations can further reduce the overall runtime whereas other hardware aspects have only limited influence. Ultimately, this allows us to process the entire NA12878 dataset in 1h 31 min, yielding an overall parallel efficiency of 53%.

Halvade is written in Java using the Hadoop MapReduce 2.0 API. The source is available at http://bioinformatics.intec.ugent.be/halvade under GPL license.

Dataset and tool versions

In all benchmarks variant calling was performed on a whole-genome DNA sequencing dataset (NA12878, human genome, Illumina Platinum Genomes) or a subset thereof. The full dataset consists of 1.5 billion 100 bp paired-end reads (50-fold coverage) stored in two 43 GB compressed (gzip) FASTQ files.

For these benchmarks, GATK version 3.1.1, BWA version 0.7.12-r1044, BEDTools version 2.17.0, elPrep version 1.0 [92015Herzeel et al.], SAMtools version 0.1.19 and Picard version 1.112 were used. The dbSNP [102001Sherry et al.] database and human genome reference found in the GATK hg19 resource bundle [112013Van der Auwera et al] were used.

Single Node Benchmarks

As the runtime of the complete NA12878 dataset on a single node is impractically high, all benchmarks in this section were performed on a representative subset of 131 million paired-end reads (about 9% of the total

number of reads). Benchmarks in this section were run on a single 24-core node (dual Intel E5-2680v3 @ 2.50GHz) with 512 GB of RAM.

Influence of the number of tasks per node

When running Halvade, the number of parallel tasks (mappers/reducers) per node can have a big influence on performance. The number of tasks per node corresponds to the number of instances of the individual tools (BWA, GATK, etc.) that are being run in parallel on a machine. One scenario is to run only a single task and to use the multithreading functionality of the tools to make use of the available cores. An alternative scenario is to run multiple tasks in parallel on the same node, each task then using only a fraction of the available cores. Because of suboptimal multithreading scalability of certain individual tools, the choice in number of tasks can have a big impact on runtime. This is illustrated in Table 1 where the runtime is shown for three scenarios: (i) 1 task using 24 cores for multithreading; (ii) 4 tasks each using 6 cores for multithreading and (iii) 24 tasks without multithreading. The sequential runtime (single core) of the pipeline is ∼30.5h. When allowing the individual tools to run 24 threads on the same machine, the runtime reduces to ∼16.5h, resulting in a very low parallel efficiency of only 7.7%. This poor scaling can be observed in both map and reduce phase, but is especially pronounced in the reduce phase. It is caused partly by the lack of multithreading support in some of the tools used, e.g. BWA sampe and Picard. However, even the modules of GATK that do support multithreading exhibit poor scaling behavior. When moving from multithreading to multitasking as supported by Halvade, runtimes decrease significantly. Using 4 tasks with 6 threads each, runtime reduces to ∼4.5 h. When using 24 tasks without multithreading a runtime of only 1h 42 min is obtained, corresponding to a parallel efficiency of 74.7%. We observed an increased CPU utilization during pipeline execution when using 24 parallel tasks compared to using multithreading in 1 task.

On this type of node, optimal runtime is achieved when using a maximum number of tasks without multithreading. However, this is only possible because the node provides a sufficient amount of RAM (512 GB in this case). Tests indicate that certain GATK modules require almost 16 GB of RAM. Therefore, the maximum number of tasks might be limited by the memory that is provided by a node.

Influence of the presence of NUMA domains

Many recent systems make use of non-uniform memory access (NUMA) domains. Each NUMA domain contains a number of CPU cores and part of the RAM. Cores have faster access to memory that resides in the same NUMA domain ('local' access) and slower access to memory that is outside this domain ('remote' access). Files on disk that are accessed by a tool are typically buffered in memory by the Linux operating system. If different processes are accessing the same file, this buffered copy of (part of) the file can be located in a different NUMA domain than that of the core accessing it. If sufficient memory

Table 1. Runtime and parallel efficiency as a function of the number of tasks per node.

	map phase		reduce phase		total	
	runtime	efficiency	runtime	efficiency	runtime	efficiency
single-threaded	14h 50 min	n/a	15h 38 min	n/a	30h 28 min	n/a
1 task × 24 threads	4h 28 min	13.84%	12h 3 min	5.41%	16h 31 min	7.69%
4 tasks × 6 threads	1h 21 min	45.78%	3h 6 min	21.01%	4h 27 min	28.53%
24 tasks × 1 threads	47 min	78.80%	55 min	71.06%	1h 42 min	74.67%

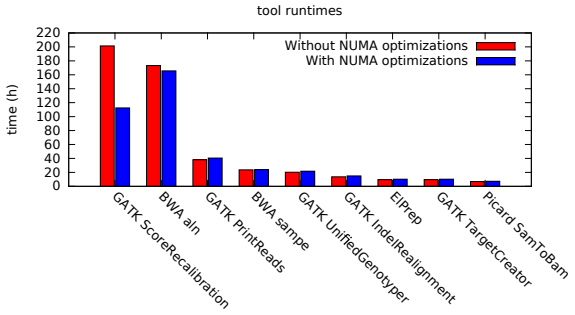

Figure 1. Comparison of the runtime (summed over all 24 parallel tasks) for each individual tool/module used in Halvade with and without optimized NUMA locality.

is available we can make distinct copies of the reference file to each of the NUMA domains and as such speed up the file access and seek times. We implemented this idea through the use of wrappers around certain Java calls. In this wrapper the NUMA domain of the assigned cores are determined and a copy is made for that domain on local scratch if it was not yet created. This way each domain has its own local copy which will be cached in the different NUMA domains.

Using 24 tasks on a single node and the entire NA12878 dataset, Fig. 1 shows the runtime of the different components (summed over all 24 tasks) of the pipeline with and without the use of the wrappers. For most components, the influence is only marginal with the ScoreRecalibrator module from GATK being a notable exception. In that particular case, a reduction in runtime of 45% can be observed when using the wrappers. This is a process where a dbSNP database file (roughly 10 GB) is intensively used to generate recalibration tables. In this case, the improved NUMA data locality considerably improves runtime.

Multi-node benchmarks

Influence of the use of solid state disks

Many tools within Halvade rely on local disk I/O (scratch). This includes reading the reference genome and accessing the dbSNP database as well as writing and

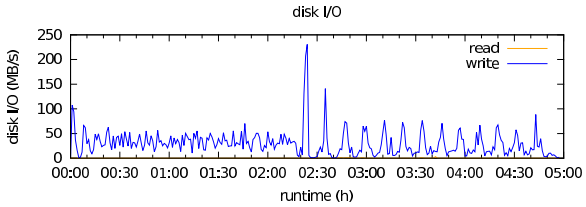

Figure 2. Disk I/O (scratch) observed on a worker node. Note that almost no data is actually being read from disk as data are still cached in memory.

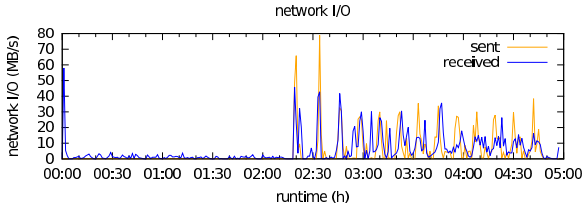

Figure 3. Network I/O observed on a single worker node.

reading intermediate data generated by the different GATK modules as well as BWA-aln and BWA-sampe. We tested the performance difference between using solid state drives (SSD) and regular hard disk drives (HDD). Test results indicate only minimal differences in runtime. This is due to the relatively low overall disk usage during the execution of the Halvade job. The disk I/O volume was measured in intervals of one minute and converted to MB/s (see Fig. 2). With the exception of a peak during sorting phase, the disk I/O is well below 100 MB/s (averaged over one minute) which is well within the range of modern HDDs. During the entire job, volumes read from disk were very low, leading us to the conclusion that almost all data written to local disk was cached in memory by the operating system and again accessed from memory in the next step.

Influence of the interconnection network

In between map and reduce phase, aligned reads are sorted according to genomic position. This parallel sorting step involves the movement of large volumes of data over the interconnection network. The network I/O volume was measured in intervals of one minute and again converted to MB/s (see Fig. 3). Again, as network I/O is below 100 MB/s, almost no performance benefit was observed by using an Infiniband interconnect over a 10 Gbit Ethernet network.

Influence of the file system

Traditionally, MapReduce relies on the Hadoop Distributed File System (HDFS) to read input and write final output data. In that case, data is stored on the local disks of the worker nodes in a distributed fashion. Alternatively, centralized

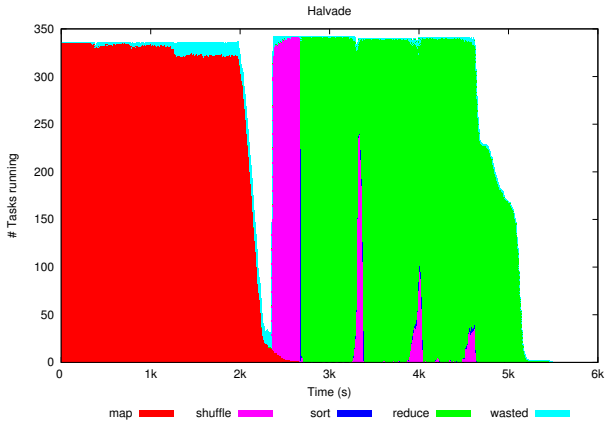

Figure 4. Execution of the NA12878 dataset on a 15-node cluster. Each node runs 24 tasks in parallel. Note that certain tasks are used by MapReduce for task tracking and scheduling purposes.

file systems such as IBM's Generalized Parallel File System (GPFS) or the Intel Enterprise Edition for Lustre software can be used. In that case, data is stored on separate data nodes and transferred to the worker nodes through an interconnection network. As the pipeline is rather compute-intensive, all three systems were able to provide data to the worker nodes at a sufficiently high rate, hence almost no performance difference was observed. However, the use of Intel's Hadoop Adapter for Lustre included in Intel Enterprise Edition for Lustre software has two advantages. First, it decreases the time spent during the sort & shuffle phase compared with HDFS/GPFS. Second, Lustre uses less memory on the worker nodes. This can be important on nodes with limited memory capacity. For instance, on nodes equipped with 64 GB of RAM running 4 Halvade tasks, we noticed that certain reduce tasks failed because of memory shortage. The cause of this is the difference in coverage over the different genomic regions and thus some tasks will have more reads to process. These reduce tasks had to be rescheduled causing an increase in runtime. On a 7-node cluster, the use of Intel's Hadoop Adapter for Lustre included in Intel Enterprise Edition for Lustre software decreased the runtime from 5h 27 min (using HDFS) to 4h 48 min on the same cluster.

Benchmark of NA12878 dataset on a 15-node cluster

Halvade was used to process the complete NA12878 dataset on a 15-node cluster, each node containing 24 CPU cores (dual-socket Intel E5-2680v3 @ 2.50GHz) with 512 GB of RAM and three solid-state drivers (SSD) of 400 GB in RAID 0 to store intermediate data (local scratch). The nodes are interconnected through an Infiniband network and access a GPFS storage through a second Infiniband network. Note that Lustre was not available on this cluster. Cloudera CDH 5.3

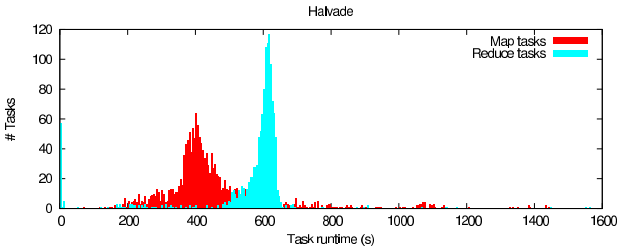

Figure 5. Distribution of runtime of the different tasks. Each map task (1569 in total) consists of aligning a chunk of ∼60 MB of the input FASTQ file to the reference genome. Each reduce task (1303 in total) involves data preprocessing and variant calling in a genomic region of about ∼2.3 Mbp.

is deployed as a Hadoop distribution by HanythingOnDemand [122014Higgs]. Halvade was configured to use 24 tasks per node, hence up to 360 tasks (24 tasks × 15 nodes) were run in parallel. NUMA optimizations were in place. On this cluster, Halvade completed read alignment and variant calling of the NA12878 dataset in 1h 31 min. Compared to a single-threaded runtime of ∼12 days, this represents and overall speedup of a factor of ∼190 or a parallel efficiency of 53%.

We can now compare this result to previously reported results in [72015Decap et al.]. A runtime of 2h 39 min was reported on a comparable 15-node cluster, however, in that case the nodes were equipped with only 64 GB of memory. Therefore, it was optimal to run only 4 parallel tasks per node instead of 24 causing significant loss of efficiency within each node. On the other hand, running 360 tasks in parallel significantly increases the task scheduling overhead and makes it more difficult for the MapReduce framework to evenly distribute the workload among the different tasks. This can be clearly observed in Fig. 4 where a non-negligible load imbalance can be observed in both the map and reduce phase. The underlying cause for this is a rather large variation in task execution time (see Fig. 5). Ultimately, with the current status of multithreading performance in the available tools, it is still best to use as many tasks on a node as possible. Note that the newer BWA-mem (also supported by Halvade) already features much improved multithreading performance over BWA-aln/sampe.

Conclusion

We investigated the impact of different hardware configurations on the runtime of Halvade, a parallel, multi-node framework that implements a variant calling pipeline according to the GATK Best Practices recommendations. Halvade relies on BWA for read mapping and GATK for variant calling.

Even though Halvade is primarily intended to allow for a multi-node parallelization of sequencing pipelines, Halvade can be used to significantly speed up post-sequencing analysis on a single node. This is because the overall parallel efficiency of the individual tools is very low: a speedup of less than 2 is observed when moving from single-threaded execution to

multithreaded execution on a 24-core machine. Part of this poor scaling behavior can be explained by the fact that BWA-sampe and Picard do not support multithreading, however, most of the GATK modules involved in the pipeline also do not exhibit good scaling behavior. This scaling behaviour has also been observed in several other mapping tools in [132013Hatem et al.]. By using Halvade on high-memory nodes, multithreading can be replaced by multitasking. The latter is far more efficient, which has also been shown in [142014Kutlu et al.], and a speedup of ∼18 is obtained on a 24-core machine.

Additionally, having much memory in a system allows to hold a copy of buffered files in each of the NUMA domains. As such, CPU cores have access to a copy in the local NUMA domain, thus avoiding remote memory access. For the GATK ScoreRecalibrator module, this improves the runtime by nearly a factor of two.

Other hardware aspects, such as local disk speed (solid state drives vs. regular hard disk drives), speed of interconnection network (Infiniband vs. Ethernet networks) or file system (HDFS vs. GPFS) have only a minor influence on overall runtime. Even though a typical whole-genome dataset involves hundreds of GB of input data and a multiple thereof of intermediate data, the sequencing pipeline is mostly compute-intensive and hence, runtime is mostly influenced by the compute capacity of a node, rather than I/O speed.

Finally, Intel Enterprise Edition for Lustre software was investigated. The use of Intel's Hadoop Adapter for Lustre included in Intel Enterprise Edition for Lustre software simplifies the shuffle & sort which leads to better performance. Additionally, Lustre uses less memory which can be important when high-memory machines are not available.

With all optimizations in place, Halvade is able to complete read alignment and variant calling of the complete NA12878 dataset in 1 hour and 31 minutes on a 15-node cluster, each node containing 24 CPU cores and 512 GB of RAM. Compared to a single-threaded runtime of ∼12 days for this pipeline, this represents an overall speedup of a factor of ∼190 or a parallel efficiency of 53%.

Acknowledgments

This work is funded by Intel, Janssen Pharmaceutica and by the Institute of the Promotion of Innovation through Science and Technology in Flanders (IWT). The computational resources (Stevin Supercomputer Infrastructure) and services used in this work were provided by the VSC (Flemish Supercomputer Center), funded by Ghent University, the Hercules Foundation and the Flemish Government department EWI. Special thanks goes to Stijn De Weirdt for his assistance with the Java wrappers to improve NUMA locality. Benchmarks on Lustre were run at the Intel Big Data Lab, Swindon, UK. We acknowledge the support of Ghent University (Multidisciplinary Research Partnership Bioinformatics: From Nucleotides to Networks).

References

[Van der Auwera et al.(2013)] Van der Auwera, G.A., Carneiro, M.O., Hartl, C., Poplin, R., del Angel, G., Levy-Moonshine, A., Jordan, T., Shakir, K., Roazen, D., thibault, J., Banks, E., Garimella, K.V., Altshuler, D., Gabriel, S., DePristo, M.A.: *From FastQ Data to High-Confidence Variant Calls: The Genome Analysis Toolkit Best Practices Pipeline.* Current Protocols in Bioinformatics. 43. 11.10.1-11.10.33 (2013)

[li et al.(2009)] Li, H., Durbin, R.: *Fast and accurate short read alignment with Burrows-Wheeler transform.* Bioinformatics, 25, 1754-1760 (2009)

[Li et al.(2009)] Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., Marth, G., Abecasis, G., Durbin, R.: *The sequence alignment/map format and SAMtools.* Bioinformatics, 25, 2078-2079 (2009)

[Wysoker et al.(2009)] Picard, http://broadinstitute.github.io/picard/

[McKenna et al.(2010)] McKenna, A., Hanna, M., Banks, E., Sivachenko, A., Cibulskis, K., Kernytsky, A., Garimella, K., Altshuler, D., Gabriel, S., Daly, M., DePristo, M.A.: *The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data.* Genome Res., 20, 1297-1303 (2010)

[Depristro et al.(2011)] Depristo, M.A., Banks, E., Poplin, R., Garimella, K.v., Maquire, J.R., Hartl, C., Philippakis, A.A., del Angel, G., Rivas, M.A., Hanna, M., McKenna, A., Fennell, T.J., Kernytsky, A.M., Sivachenko, A.Y., Cibulskis, K., Gabriel, S.B., Altshuler, D., Daly, M.J.: *A framework for variation discovery and genotyping using next-generation DNA sequencing data.* Nature Genetics, 43, 491-498 (2011)

[Decap et al.(2015)] Decap, D., Reumers, J., Herzeel, C., Costanza, P., Fostier, J.: *Halvade: scalable sequence analysis with MapReduce*, Bioinformatics, in press (2015)

[Dean et al.(2008)] Dean, J., Ghemawat, S.: *MapReduce: simplified data processing on large clusters.* Commun. ACM, 51, 107-113 (2008)

[Herzeel et al.(2015)] Herzeel, C., Costanza, P., Decap, D., Fostier, J., Reumers, J.: *elPrep: High-Performance Preparation of Sequence Alignment/Map Files for Variant Calling* PLoS ONE, 10(7) (2015)

[Sherry et al.(2001)] Sherry, S.T., Ward, M.H., Kholodov, M., Phan, L., Smigielsky, E.M., Sirotkin, K.: *dbSNP: the NCBI database of genetic variation.* Nucleic Acids Research, 29(1), 308-311 (2001)

[Van der Auwera et al(2013)] GATK resource bundle, ftp://gsapubftp-anonymous@ftp.broadinstitute.org/bundle/2.8/hg19

[Higgs(2014)] HanythingOnDemand, https://github.com/hpcugent/hanythingondemand

[Hatem et al.(2013)] Hatem, A., Bozda, D., Toland, A.E., atalyrek, .V.: *Benchmarking short sequence mapping tools*, BMC Bioinformatics, 14, 184 (2013)

[Kutlu et al.(2014)] Kutlu, M., Agrawal, G.: *PAGE: A Framework for Easy PArallelization of GEnomic Applications*, IPDPS (2014)