RNACache: Fast Mapping of RNA-Seq Reads to Transcriptomes using MinHashing

Julian Cascitti, Stefan Niebler, André Müller, and Bertil Schmidt

Johannes Gutenberg-Universität Mainz
Institute of Computer Science, 55128 Mainz, Germany
jcascitt@students.uni-mainz.de,
{stnieble, muellan, bertil.schmidt}@uni-mainz.de

Abstract. The alignment of reads to a transcriptome is an important initial step in a variety of bioinformatics RNA-seq pipelines. As traditional alignment-based tools suffer from high runtimes, alternative, alignment-free methods have recently gained increasing importance. We present a novel approach to the detection of local similarities between transcriptomes and RNA-seq reads based on context-aware minhashing. We introduce RNACache, a three-step processing pipeline consisting of minhashing of k-mers, match-based (online) filtering, and coverage-based filtering in order to identify truly expressed transcript isoforms. Our performance evaluation shows that RNACache produces transcriptomic mappings of high accuracy that include significantly fewer erroneous matches compared to the state-of-the-art tools RapMap, Salmon, and Kallisto. Furthermore, it offers scalable and highly competitive runtime performance at low memory consumption on common multi-core workstations. RNACache is publicly available at:
https://github.com/jcasc/rnacache

Keywords: Bioinformatics · Next-generation sequencing · RNA-seq · Transcriptomics · Read mapping · Hashing · Parallelism · Big Data

1 Introduction

Obtaining data from the sequencing of RNA (RNA-seq) is a major advancement in medical and biological sciences that allows for the measurement of gene expression. It has thus become an important technique for gaining knowledge in a wide range of applications including drug development and understanding of diseases [23]. The increasing availability and large size of next generation sequencing (NGS) data has also established the need for highly optimized big data methods to align (or map) the produced reads to reference sequences [20]. Since it is estimated that over hundreds of millions of human genomes and transcriptomes will be sequenced by 2025, finding fast RNA-seq mapping algorithms is of high importance to research [22].

Classical *read mapping* aims to identify the best alignment(s) of each read to a given reference genome. In contrast, RNA-seq often requires the mapping

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_31

of each produced read to a reference transcriptome, consisting of a collection of genes, where each gene is represented by several alternative transcripts (known as *isoforms*). Due to alternative splicing, many isoforms of the same gene can be transcribed, which often contain highly similar subsequences. Thus, unlike mapping of genomic NGS data, where redundancies are usually less common, highly similar regions between isoforms and homologs have to be taken into account when mapping RNA-seq data. Because the number of potential origins tends to be high, the process of determining the true origin of each sequencing read becomes complex. This, in turn, can result in inaccurate expression estimation of transcripts if not accounted for [6].

From an algorithmic perspective, established short-read aligners such as BWA-MEM [11] and Bowtie2 [9] are based on *seed-and-extend* approaches which map sequencing reads to a reference genome by first identifying seeds using FM-index based data structures. Seeds are extended (e.g., by using fast versions of dynamic programming based alignment algorithms) in order to verify whether a seed is actually contained in a full alignment. Unfortunately, the computation of traditional alignments for a large number of reads typically exhibits long run times. The problem is further exacerbated by the multiplicity of similar potential origins in transcriptome data, which can entail high numbers of seeds.

Consequently, a number of methods have recently been introduced for fast mapping of RNA-seq reads to reference transcriptomes. These approaches are based on the concepts of *lightweight-alignment* or *quasi-mapping* which rely on the observation that the task of identifying the most likely isoform(s) each read may originate from does not require precise and expensive alignment computation. RapMap [21] works by locating consecutive k-mers within a suffix array of concatenated target transcripts and finding the longest possible substring shared between read and target. Sailfish [18] and Selective Alignment (RapMapSA) [19] both extend RapMap by adding a number of criteria and filters. Kallisto [2] generates pseudoalignments by determining the set of transcripts containing all of the read's k-mers using a de-Bruijn-graph. While these approaches clearly outperform traditional aligners such as STAR [5] in terms of runtime, the number of reported potential read origins (called *hits-per-read*) can be high. Furthermore, scalability (with respect to taking advantage of an increasing number of cores of modern CPUs) is often limited while memory consumption can be high.

In this paper, we present RNACache – a new algorithm for fast and memory-efficient mapping of RNA-seq reads to a given reference transcriptome. Our method utilizes *minhashing*, a technique [3] known from processing big data to map reads based on an approximate Jaccard-Index by analyzing k-mer subsamples [10]. Subsequently, a selection of task-specific filters is applied utilizing statistical information gathered during the mapping of all reads in order to significantly reduce the number of reported hits-per-read compared to previous approaches while still maintaining high recall. Furthermore, we take advantage of multi-threading to design a high-speed yet memory-efficient implementation that can exploit large number of cores available in modern workstations. Our performance evaluation shows that RNACache produces a lower number of hits-

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_31

per-read than RapMap, RapMapSA, Salmon, and Kallisto while achieving higher recall at the same time. In addition, RNACache is able to outperform all other tested mapping-based tools in terms of runtime and memory consumption.

2 Method

We adopt minhashing – a locality sensitive hashing (LSH) based data subsampling technique. It has been successfully employed by search engines to detect near duplicate web pages [4] but has recently gained popularity in NGS analysis with example applications including genome assembly [1], sequence clustering [16], metagenomics [13], food sequencing [8], and single-cell sequencing [15]. To our knowledge, we are the first to apply this concept in the context of RNA-seq mapping.

We apply minhashing for memory-efficient transcriptome database (hash-map) construction and querying. Query outputs are used for initial read assignments. These are further refined by two filters (online and coverage filter) in order to derive final read classifications. The workflow of RNACache is illustrated in Figure 1.

2.1 Database Construction

In order to construct an index, reference transcriptome sequences (also called *transcripts* or *targets*) are covered in slightly overlapping (default: $k-1$ nucleotides) *windows*. From each window a preset maximum number of k-mers (substrings of length k) possessing the lowest hash values (referred to as *features*) are selected to form what is called a *sketch*. A hash-map is constructed that maps every *feature* to its corresponding locations, i.e., a list of transcript and window identifiers in which it occurs. Strandedness is handled by substituting every k-mer with the lexicographical minimum of itself and its reverse-complement, referred to as its *canonical* k-mer, prior to hashing.

2.2 Querying and Initial Read Assignment

When querying a read, it is *sketched* similarly to a reference sequence and a lookup of each of its features is performed on the index yielding a list of reference transcript locations which share features with the read, also referred to as *hits* or *matches*. In this way, we determine which windows of which targets share features, respectively k-mers, with the queried read. The number of features shared by the read and a single target window can be interpreted as an approximation of the Jaccard-Index of their respective k-mer sets, i.e., as a measure of their similarity and hence of a local similarity between corresponding target transcriptome sequence and read.

In order to determine larger regions of similarity, a *candidate* region is subsequently selected from each reference transcriptome sequence by determining the *contiguous-window-range* of a maximum preset length spanning the most feature

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_31

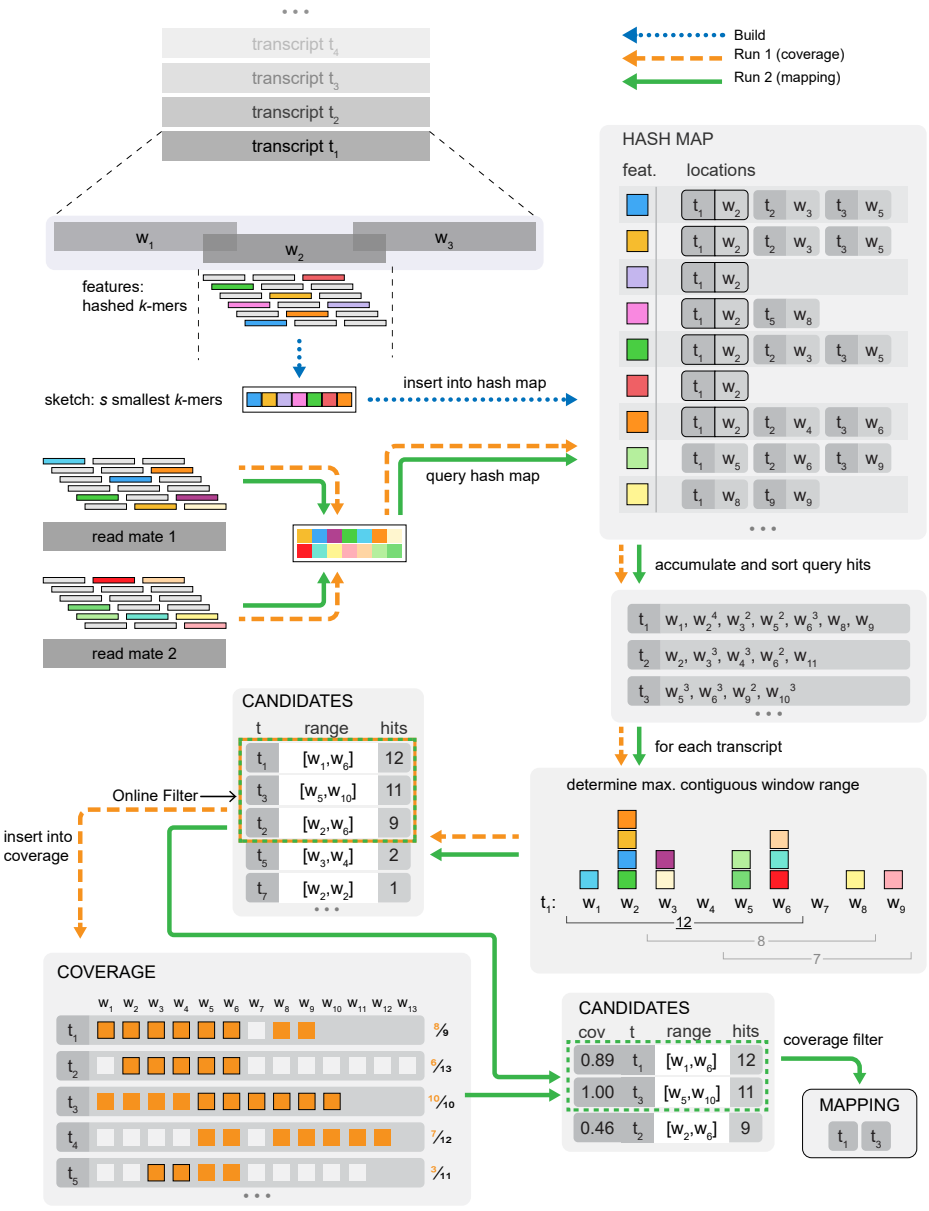

Fig. 1: Workflow of RNA-Cache. Build: Transcripts (t_j) are partitioned into windows (w_i). The s smallest features (hashes of canonical k-mers) of each window are computed and inserted into the hash map (database). Run 1 and Run 2: The hash map is queried with the s smallest features of each read. The returned hits are accumulated per transcript and candidates identified by determining the contiguous window range spanning the most hits on each transcript. After online-filtering, candidates are inserted into the coverage list, which is used to filter candidates in Run 2 (mapping) in order to determine the final mapping.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_31

hits. This maximum window range corresponds to the maximum expected *insert size*, i.e. fragment length for paired-end reads. Thus, *candidates* can be described as tuples of the form $(tgt, (b, e), hits)$, where tgt is an ID of the target sequence, (b, e) is a tuple of window IDs, spanning the closed interval $[b, e]$ of windows on the target sequence, and $hits$ is the number of features shared by the target window interval and the read.

Note that, features which occur in a very high number of locations (default: more than 254) are considered uninformative and are therefore deleted from the index. As these features correspond to k-mers appearing in many different reference sequences, they offer little discriminatory information regarding the origin of a queried read.

After querying, RNACache applies an online filter (see Section 2.3) and a subsequent post-processing (coverage) filter (see Section 2.4) to select the most plausible transcript(s) among them, creating the final mapping of the read set. The mapping generated in this way seeks to minimize the total number of candidates per read while maximizing the ratio of reads to which the correct original transcript is assigned (*recall*).

2.3 Online Filtering

Candidate targets which a considered read might originate from are refined by an online filtering process based on absolute and relative hit counts. Consider a read r's set of candidates $C(r) = \left\{ c_1 = (tgt_1, (b_1, e_1), hits_1),\ c_2, \dots \right\}$ (see Section 2.2). Online filtering will permit the candidate set

$$F_o(C(r)) = \left\{ \Big(tgt, (b, e), hits \Big) \in C(r) \;\middle|\; hits > t^{min}, \frac{hits}{hits^{max}} > t^{cutoff} \right\} \quad (1)$$

where $hits^{max} = \max_{c_k \in C(r)} hits_k$.

Informally, this filter permits only those candidates whose number of hits exceeds t^{min} in absolute terms and is greater than a fraction of t^{cutoff} relative to the highest number of hits of any candidate of the same read. The second relative filter has the advantage of innately scaling with other parameters and properties of the input and reference data which affect the number of hits produced by well-fitting reads.

2.4 Post-processing (Coverage) Filter

After online filtering, some reads can still map equally well to different targets that share highly similar subsequences. However, it is unlikely that all of these potential transcripts of origin are actually expressed in a transcriptomic sample. Thus, a post processing filter is applied in order to distinguish between expressed and non-expressed isoforms containing highly similar subsequence regions. For this filter it is necessary to consider the complete set of reads as a whole, rather than filter candidates solely on a per-read basis (as done by the online filter).

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_31

RNA-seq data sets typically have a large coverage [14]. Thus, we can assume with high confidence that every window of every transcript actually expressed in the sample will appear as a match of at least some reads during the querying process. By keeping track of all target windows *covered* in this way, we can identify target regions with no matching reads, and discard the target as unlikely to be expressed in the sample. Thus, we can use the information gained by reads mapping to a sequence's unique regions, or alternatively the lack thereof, to reason about the correct mapping of reads to non-unique target regions. For this purpose, we use the concept of *coverage* as follows.

Coverage. We define the *coverage* of a target as the ratio of its number of *covered* windows to its number of total windows. A window is said to be *covered* if the following two conditions hold:

1. It is an element of the *contiguous window range* of a *candidate* of at least one read.
2. It shares at least one feature with that read.

Formally, we define $\mathfrak{R}$ as the set of all input reads and $W(tgt)$ as the set of windows of target tgt. Then, the *covered window set* of tgt is defined as:

$$covered(tgt) = \Big\{ w_{tgt,j} \in W(tgt) \;\Big|\; \exists r \in \mathfrak{R} : \exists (tgt, (b,e), hits) \in F_o(C(r)) : j \in [b,e] \underbrace{\wedge\; S(w_{tgt,j}) \cap S(r) \neq \emptyset}_{Condition\ 2} \Big\}, \tag{2}$$

where $w_{tgt,j} \in W(tgt)$ is the j-th window of target tgt while $S(w_{tgt,j})$ and $S(r)$ denote the *sketch* of $w_{tgt,j}$ and r respectively.

Furthermore, we define the *coverage statistic* of a target tgt as:

$$cov(tgt) = \frac{|covered(tgt)|}{|W(tgt)|}. \tag{3}$$

In the following, the term *coverage* refers to either a target's *covered window set* or *coverage statistic*, unless the distinction is relevant.

Condition 2 ensures that only those windows are counted towards a target's coverage which contribute to the candidate's match count, i.e., share features with the read. This is primarily relevant in the case of paired-end reads where it reduces false positives in the coverage caused by the inclusion of regions that differ from the sequenced fragment in the region located between the read mates. Condition 2 can also be omitted by using a runtime parameter to handle weakly expressed input data sets or transcriptomes that contain transcripts that are too short for their center regions to appear in paired-end reads.

Coverage-based Filtering. The post-processing coverage filter admits those candidates whose target's coverage statistic exceeds a fraction t^{cov} of the highest coverage statistic of any candidate of the same read.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_31

Formally:

$$F_{cov}(F_o(C(r))) = \left\{ c = (tgt, (b, e), hits) \;\middle|\; c \in F_o(C(r)), \frac{cov(tgt)}{\max\limits_{c_k \in F_o(C(r))} cov(tgt_k)} \geq t^{cov} \right\} \tag{4}$$

Per-read maximum-normalization allows the filter to automatically scale with properties of the read and reference transcriptome as well as other parameters that affect overall coverage of the dataset and has empirically proven to increase mapping accuracy compared to a simple coverage threshold. Additionally, it has the property of never rejecting the most covered candidate of any read, meaning that it cannot by itself cause a read to remain entirely unmapped. However, the user can disable this normalization in favor of a simple coverage threshold, if desired.

Coverage Data Structure. In order to keep track of coverage, an additional data structure in form of a hash map is constructed while reads are queried, in which targets are mapped to a list of their covered windows, based on the aforementioned definition and the *candidates* resulting from online filtering. Subsequent application of the post-processing filter would require all candidates of each read to be stored. For typical RNA-seq read set sizes, this can become infeasible; e.g. the required memory for a medium-sized set of $\approx 50M$ reads and ≈ 5 candidates per read can already be expected to be on the order of 2.5GB. Different parameters regarding online filtering and very large read sets can quickly increase this requirement by at least an order-of-magnitude. Thus, the default behavior of RNACache is not to store candidates. Instead, the sketching and querying process, including online-filtering, is repeated after the coverage data structure has been completely assembled.

2.5 Parallelization

Input sequences are processed by multiple CPU threads in parallel that communicate work items with the help of a concurrent queue.

The database build phase uses three threads. One thread reads transcripts from the input files, one sketches sequences into hash signatures and one inserts sketches into the hash table. Using more than three threads does not improve performance since our hash table does not allow for concurrent insertion.

The query phase uses one producer thread for reading the input RNA-seq reads from file and multiple consumer threads to accelerate the time-consuming classification of reads. The producer thread places batches of input reads (default is 4096 per batch) in the queue from which consumer threads then extract them for processing. The individual results of each consumer thread are successively merged into global data structures and/or written to a results file.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_31

3 Evaluation

We assessed the performance of RNACache in terms of mapping accuracy and speed using simulated and real RNA-seq data, and compared it to a number of existing state-of-the-art tools performing transcriptomic mappings without full alignments:

- RapMap v.0.6.0 [21],
- RapMapSA v.0.6.0 [19],
- Kallisto v.0.46.2 [2],
- Salmon v.1.2.1 [17],

as well as to Bowtie2 v.2.4.1 [9], a commonly used full alignment tool.

Simulated datasets were generated using Flux simulator (v.1.2.1) [7] by applying appropriate parameters for human transcriptome RNA-seq experiments. The utilized reference transcriptome consists of 100,566 transcripts of protein-coding genes, taken from GENCODE Human Release 34 (GRCh38.p13). Reads were generated from this transcriptome using corresponding annotations from the same release.

Our tests were conducted on a workstation with a dual 22-core Intel(R) Xeon(R) Gold 6238 CPU (i.e. 88 logical cores total), 187 GiB DDR4 RAM (NUMA), two PC601 NVMe SK Hynix 1TB in RAID0 running Ubuntu 18.04.4.

With the exception of Salmon, whose binary release was used directly, all tools were built using g++ v.7.5.0. As all considered tools other than RNACache and Kallisto use SAM format output, Samtools [12] was utilized to convert their output into BAM format on-the-fly.

3.1 Accuracy Evaluation using Simulated Reads

To assess accuracy, datasets of $\approx$ 48 million paired-end reads of length 2×76 base-pairs were generated. Mapping quality is assessed in terms of three measures:

- *Recall*, defined as the fraction of total input reads whose mapping result includes the true origin,
- *Hits-per-read* or *HPR*, defined as the mean number of returned candidates of a mapped read, i.e. disregarding unmapped reads,
- *True-hit-rate* or *THR*, which is the ratio of the number of true origins found and the total number of candidates assigned.

This form of evaluation is based on the evaluation of RapMap [21], in which a read is considered a "True Positive" if its mapping includes its true origin, regardless of the number of other candidates. Furthermore, *HPR* and *THR* can be considered a conceptual counterpart to the *precision* quality in binary classification tasks.

Ground truth mappings were extracted from read headers generated by the simulator. Bowtie2 has been included to allow for a comparison to an established full alignment-based tool. Note that it uses a "k" parameter specifying the maximum number of distinct, valid alignments it will search for. As this setting has

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_31

a direct effect on the HPR statistic, Bowtie2 was executed with three different settings: $k = 1$ (def.), $k = 20$, $k = 200$.

Table 1 shows the results whereby the experiments were repeated five times with different read data and the reported values are averages. In addition to recall, HPR, and THR, the percentage of reads that have been mapped to at least one transcript is provided in the first row ("aligned").

Table 1: Mapping results for simulated reads (bold denotes best value).

	RNACache	RapMap	RapMapSA	Salmon	Kallisto	Bowtie2		
						$k = 1$	$k = 20$	$k = 200$
Aligned (%)	**98.2**	97.9	83.7	86.5	91.8	96.4	96.9	97.5
Recall (%)	**96.4**	96.0	83.4	86.1	89.4	41.2	95.0	96.0
HPR	3.9	5.0	5.0	5.0	4.6	**1.0**	4.8	5.8
THR	25.2	19.7	20.0	19.9	21.3	**42.7**	20.5	17.1

RNACache achieves the highest percentage of aligned reads and highest recall, while it is second best for HPR and THR. It is able to assign the correct original transcript to 96.4% of all reads and produces fewer than 4 mappings per read on average. While RapMap comes close in terms of recall, it produces more than 25% more false transcripts assignments ($HPR = 5.0$). RapMapSA and Salmon fail to map large portions of reads. Kallisto has similarly low recall. Bowtie2 with $k = 1$ achieves the best HPR and THR at the expense of the worst recall (only $\approx 40\%$). The results of Bowtie2 with medium setting ($k = 20$) are in the vicinity of the other tools while being balanced more towards lower HPR/recall. Even Bowtie2 with highest k-value ($k = 200$) is not capable of beating RNACache in terms of recall, while accumulating the largest amount of false matches (HPR ≈ 5.8).

Impact of Filtering. In order to analyze the effectiveness of our introduced filters, we evaluated their impact to the RNACache processing pipeline in terms of the percentage of aligned reads, recall, HPR, and THR. The results in Table 2 show that the application of the online filter refines the initial read-to-reference mappings by improving both HPR and THR by over an order-of-magnitude while only slightly decreasing recall. The additional coverage filter can improve HPR and THR even further at only a minor decrease of recall.

3.2 Runtime and Parallel Scalability

In addition to accuracy, runtimes were measured in a standard usage scenario. All tools were executed with 88 threads in total on our test system. Query runtimes are shown in Figure 2. Breakdown of absolute runtimes in build (index) and query phase (including disk I/O) as well as peak memory consumption are

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_31

Table 2: Impact of filtering steps on the mapping results of RNACache.

	No filter	Online	Online + Coverage
Aligned (%)	99.0	98.2	98.2
Recall (%)	97.6	96.9	96.4
HPR	60.2	5.3	3.9
THR	1.6	18.7	25.2

provided in Table 3. With the exceptions of Kallisto and RNACache, outputs were piped into Samtools for live SAM-to-BAM conversion, for which 8 threads (4 in the case of Bowtie2) were deducted and assigned to Samtools [12]. This represents a common toolchain in bioinformatics when working with the involved tools, as handling large SAM output files is impractical.

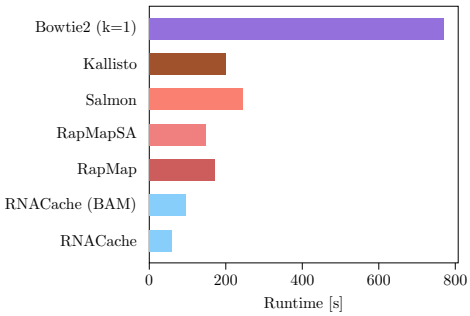

Fig. 2: Query runtimes on synthetic data including SAM-to-BAM conversion and hard-disk I/O.

RNACache outperforms all other tools in terms of runtime, in both its default and BAM output modes. The minhash-based hash map construction (build phase) of RNACache is faster than the corresponding index data structure construction of the other methods; i.e. it outperforms suffix array construction used by RapMap, RapMapSA, and Salmon by at least 1.9×, deBruijn Graph construction used by Kallisto by 2.1×, and FM-index construction used by Bowtie2 by 16.9×. In its default output mode, the query phase of RNACache achieves speedups of 2.9, 2.5, 4.2, 3.4, and 13.2 compared to RapMap, RapMapSA, Salmon, Kallisto, and Bowtie2 (best case, $k = 1$), respectively. Futhermore, RNACache exhibits a low memory footprint.

An advantage of RNACache is that it provides its own significantly more compact output format in which only the identifiers of the query read and of its mapped targets, as well as the number of shared minhashing features, are stored. Optionally, the mapping can be output in the BAM format, which is more suited

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_31

Table 3: Build (index creation) / Query runtimes and peak memory (RAM) consumption in a common SAM/BAM conversion chain, including conversion and disk I/O (best values in bold).

	RNACache		RapMap		Salmon	Kallisto	Bowtie2		
	def.	BAM	def.	SA			$k=1$	$k=20$	$k=200$
build [s]	**54.8**	**54.8**	102.5	102.5	117.9	188.2	931.0	931.0	931.0
query [s]	**58.4**	96.1	170.5	148.3	244.7	199.6	769.2	1968.9	5976.2
memory [GB]	2.6	8.5	107.0	107.0	18.3	16.3	**1.5**	3.7	21.4

for alignment-based methods. While this format offers compatibility with many downstream tools expecting traditional alignments, none of the tested alignment-free methods actually produce sufficient information to utilize all of the format's columns to their full extent in the way alignment-based methods can.

To evaluate parallel thread scalability of the compared mapping algorithms, we benchmarked all tools using various thread counts with minimal disk I/O and without subsequent format conversions. Note that for Bowtie2 thread counts lower than 8 were not used for the setting of $k = 200$, as corresponding runtimes would be exceedingly long. Disk I/O was minimized by ensuring that all data was preloaded into page cache beforehand and output was suppressed to the extent possible. The benchmark results are shown in Figure 3. Data sets of ≈ 20 mio. reads were used, benchmarks were repeated three times and results averaged.

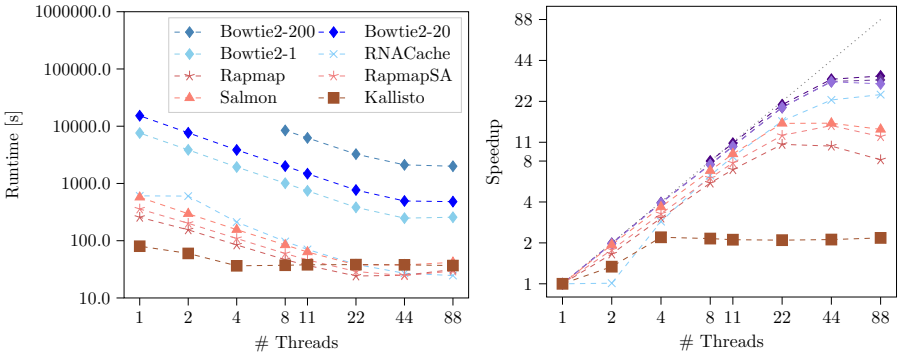

Fig. 3: Query runtimes and speedups of all tested tools for varying numbers of CPU threads using synthetic data. (Suppressed I/O, no format conversion.)

Bowtie2 exhibits the highest parallel efficiency (measured in terms of speedup divided by the number of utilized threads) for large numbers of threads but still has the longest overall runtime. The mapping-based methods (RNACache, RapMap, RapMapSA, Salmon, Kallisto) generally offer superior runtime perfor-

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_31

mance compared to Bowtie2 but exhibit lower parallel efficiency. RNACache outperforms all other mapping-based tools when executed with high thread counts and exhibits superior parallel scalability when more than 22 threads are used.

The noticeable drop in parallel efficiency for more than 22 threads is most likely related to the 2-socket NUMA architecture of our benchmarking system. Additionally, we can observe that hyper-threading (which is used when going from 44 to 88 threads) is not always beneficial when physical cores are exhausted, increasing runtimes due to higher scheduling and inter-thread communication overhead in some cases.

Our results suggest that a minhashing-based method can offer better scalability and lower memory consumption than other state-of-the-art mapping algorithms, and thus might lend itself to an effective port to many-core architectures such as CUDA-enabled GPUs.

3.3 Evaluation using Experimental Data

As ground truth data is usually not available for experimental data, our evaluation on real RNA-seq data was performed by assessing the concordance of RNACache to other tools. We have used the NCBI GEO accession SRR1293902 sample consisting of 26M 75bp paired-end reads (Illumina HiSeq) and the same reference transcriptome as in Section 3.1. For each read r_i, we have compared the set of transcripts $M_{RNACache}(r_i) = \{t_{j_1}, t_{j_2}, ...\}$ assigned to it by RNACache, to the sets $M_{kallisto}(r_i), ..., M_{RapMap}(r_i)$ returned by other tools by accumulating the occurrences of various set relations $(\subset, \supset, =, ...)$ per tool. The results are shown in Figure 4.

We can observe that RNACache shares over half of all mappings with each tool, with the exception of Bowtie2's default setting, which limits the size of paired alignments to one. In comparison with most tools, RNACache assigns an equal or subset mapping to a large majority of reads ($\approx 80\%$), in line with the already observed hits-per-read values, while leaving fewer reads unmapped. Kallisto constitutes an exception among mapping-based tools in that its stricter rejection of sub-optimal alignments leads to RNACache assigning a larger number of reads to superset mappings. Furthermore, only a vanishingly small number of reads left unmapped by RNACache were mapped by any other tool.

4 Conclusion

We have presented RNACache, a minhashing-based method for determining local sequence similarities by analyzing intersections of local pseudo-random k-mer subsets and applied it to the important task of transcriptomic mapping of RNA-seq data. Using a multi-step filtering process utilizing read-global transcript coverage information, our approach is able to refine initial read-to-target mappings in order to produce transcriptomic mappings of high accuracy. Our performance evaluation shows that RNACache is able to outperform the state-of-the-art mapping tools RapMap, RapMapSA, Salmon, and Kallisto in terms of

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_31

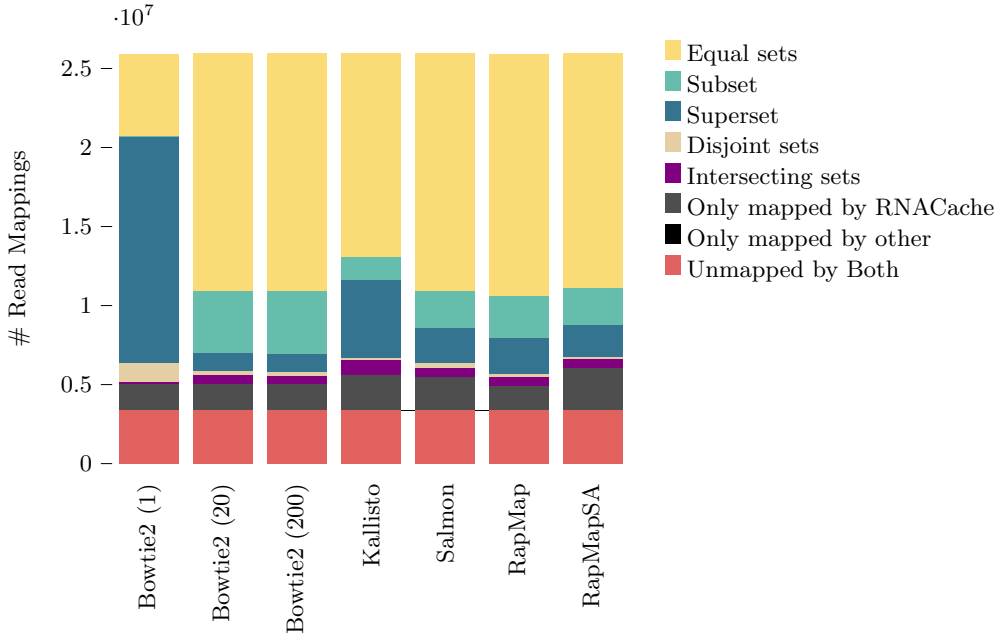

Fig. 4: Number of reads whose mapping by RNACache is either: equal, a subset, a superset, a disjoint set, an intersecting set in comparison to the other tool's mapping, a non-empty set where the other tool's mapping is empty, or an empty set whereas the other tools' mapping is non-empty, or also empty.

recall, hits-per-read, and true-hit-rate while having the lowest memory footprint. Furthermore, it offers the fastest runtimes among the tested tools on common multi-core workstations. RNACache and the scripts and parameters used in its evaluation are publicly available at https://github.com/jcasc/rnacache .

References

1. Berlin, K., Koren, S., Chin, C.S., et al.: Assembling large genomes with single-molecule sequencing and locality-sensitive hashing. Nat Biotech **33** (2015)
2. Bray, N.L., Pimentel, H., Melsted, P., Pachter, L.: Near-optimal probabilistic RNA-seq quantification. Nature Biotechnology **34**(5), 525–527 (Apr 2016)
3. Broder, A.Z.: On the resemblance and containment of documents. In: Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No.97TB100171). pp. 21–29 (1997)
4. Broder, A.Z.: Identifying and filtering near-duplicate documents. In: Proceedings of the 11th Annual Symposium on Combinatorial Pattern Matching. p. 1–10. COM '00, Springer-Verlag, Berlin, Heidelberg (2000)

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_31

5. Dobin, A., Davis, C.A., Schlesinger, F., Drenkow, J., Zaleski, C., Jha, S., Batut, P., Chaisson, M., Gingeras, T.R.: Star: ultrafast universal rna-seq aligner. Bioinformatics **29**(1), 15–21 (2013)
6. Garber, M., Grabherr, M.G., Guttman, M., Trapnell, C.: Computational methods for transcriptome annotation and quantification using rna-seq. Nature methods **8**(6), 469–477 (2011)
7. Griebel, T., Zacher, B., Ribeca, P., Raineri, E., Lacroix, V., Guigó, R., Sammeth, M.: Modelling and simulating generic RNA-Seq experiments with the flux simulator. Nucleic Acids Research **40**(20), 10073–10083 (09 2012)
8. Kobus, R., Abuín, J.M., Müller, A., Hellmann, S.L., Pichel, J.C., Pena, T.F., Hildebrandt, A., Hankeln, T., Schmidt, B.: A big data approach to metagenomics for all-food-sequencing. BMC bioinformatics **21**(1), 1–15 (2020)
9. Langmead, B., Salzberg, S.L.: Fast gapped-read alignment with bowtie 2. Nature Methods **9**(4), 357–359 (Apr 2012)
10. Leskovec, J., Rajaraman, A., Ullman, J.D.: Mining of massive data sets. Cambridge university press (2020)
11. Li, H.: Aligning sequence reads, clone sequences and assembly contigs with bwa-mem (2013)
12. Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., Marth, G., Abecasis, G., Durbin, R., Subgroup, .G.P.D.P.: The Sequence Alignment/Map format and SAMtools. Bioinformatics **25**(16), 2078–2079 (06 2009)
13. Müller, A., Hundt, C., Hildebrandt, A., et al.: Metacache: context-aware classification of metagenomic reads using minhashing. Bioinf. **33**(23) (2017)
14. Nellore, A., Collado-Torres, L., Jaffe, A.E., Alquicira-Hernández, J., Wilks, C., Pritt, J., Morton, J., Leek, J.T., Langmead, B.: Rail-rna: scalable analysis of rna-seq splicing and coverage. Bioinformatics **33**(24), 4033–4040 (2017)
15. Niebler, S., Müller, A., Hankeln, T., Schmidt, B.: Raindrop: Rapid activation matrix computation for droplet-based single-cell rna-seq reads. BMC bioinformatics **21**(1), 1–14 (2020)
16. Ondov, B.D., Treangen, T.J., Melsted, P., et al.: Mash: fast genome and metagenome distance estimation using minhash. Genome Biology **17:132** (2016)
17. Patro, R., Duggal, G., Love, M.I., Irizarry, R.A., Kingsford, C.: Salmon provides fast and bias-aware quantification of transcript expression. Nature Methods **14**(4), 417–419 (Mar 2017)
18. Patro, R., Mount, S.M., Kingsford, C.: Sailfish enables alignment-free isoform quantification from RNA-seq reads using lightweight algorithms. Nature Biotechnology **32**(5), 462–464 (Apr 2014)
19. Sarkar, H., Zakeri, M., Malik, L., Patro, R.: Towards selective-alignment: Bridging the accuracy gap between alignment-based and alignment-free transcript quantification. In: Proc. 2018 ACM Int. Conference on Bioinformatics, Computational Biology, and Health Informatics. p. 27–36. BCB '18, ACM (2018)
20. Schmidt, B., Hildebrandt, A.: Next-generation sequencing: big data meets high performance computing. Drug discovery today **22**(4), 712–717 (2017)
21. Srivastava, A., Sarkar, H., Gupta, N., Patro, R.: RapMap: a rapid, sensitive and accurate tool for mapping RNA-seq reads to transcriptomes. Bioinformatics **32**(12), i192–i200 (06 2016)
22. Stephens, Z.D., Lee, S.Y., Faghri, F., Campbell, R.H., Zhai, C., Efron, M.J., Iyer, R., Schatz, M.C., Sinha, S., Robinson, G.E.: Big data: astronomical or genomical? PLoS biology **13**(7), e1002195 (2015)
23. Wang, Z., Gerstein, M., Snyder, M.: Rna-seq: a revolutionary tool for transcriptomics. Nature reviews genetics **10**(1), 57–63 (2009)

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_31