White Paper

White paper on CLC read mapper

October 10, 2012

Sample to Insight

QIAGEN Aarhus · Silkeborgvej 2 · Prismet · 8000 Aarhus C · Denmark
Telephone: +45 70 22 32 44 · www.qiagenbioinformatics.com · support-clcbio@qiagen.com

Contents

1 Abstract

After decades of biological research relying on Sanger sequencing, massively parallel high throughput sequencing (HTS) techniques have created a broad range of new and exciting research applications by increasing the output sequencing data dramatically. Although allowing for hitherto unseen DNA and RNA sequencing and binding study designs, HTS technologies have created remarkable bioinformatic challenges. In typical resequencing studies *read mappers* are used to align sequence reads to reference genomes. Read mapping, while not too time consuming in the Sanger-century, quickly evolved into one of the most insistent bottlenecks. Nowadays, researchers can resort to a broad range of read mapping solutions and it becomes increasingly challenging to choose software that optimally meets given requirements. Most demanding characteristics of read mappers include accuracy of the underlying read mapping algorithms and computational costs of running them. But since read mapping is not any more the expert task it was, Biologists have signaled strong interest in running analyses by themselves, such that ease of use displays a typical requirement, too.

We here present a thorough study of the read mapping solutions included in all *CLC Genomics Workbench*, *CLC Genomics Server* and *CLC Assembly Cell* products. We demonstrate convenient use and prove both market-leading performance and accuracy in various read mapping scenarios. Benchmarks exercised show in detail that the *CLC* read mapper not only maps more than 1.3 billion *Illumina* reads (100Nt, paired-end) in less than 5 hours, but also consistently achieves competitive mapping accuracy even for complex read data, such these originating from the *PacBioRS* system. We conclude that the *CLC* read mapper consistently performs neck and neck with the market in all major disciplines and thus provides a strong basis for a broad range of resequencing applications.

2 Introduction

In the mid-1970s Sanger and Maxam described a DNA-sequencing technology [Maxam and Gilbert, 1977, Sanger et al., 1977] termed *Sanger-sequencing*, that enabled for a broad range of research topics for molecular biologists. Ever since, costs of obtaining DNA-sequences have decreased steadily, but still displayed a major limitation. Derivations of high throughput sequencing (HTS) technologies recently caused dramatic cost-reductions and thus created an even broader range of exciting sequencing applications [Bosch and Grody, 2008]. And while sequencing platform vendors engaged themselves in a race for the 1000$ human genome [Service, 2006], output volumes of their HTS instruments kept stalling most hitherto bioinformatic tools and pipelines. Bioinformaticians and algorithmicians have consequently revised their strategy from developing accurate solutions to developing solutions that can cope with the data volumes in the first place, to overcome momentous bottlenecks present in many sequencing scenarios. Mapping sequence reads to their respective reference genome is one of those bottlenecks and has evolved to a major of its own.

The traditional Smith-Waterman algorithm (SW) [Smith and Waterman, 1981] is well described to always find the optimal alignment between a pair of sequences and is thus considered the most accurate alignment algorithm available. However, the algorithmic complexity of SW is too high to efficiently align millions of sequences to large reference genomes, such as the human or the murine. In contrast, heuristic algorithms, such as BLAST [Altschul et al., 1990] and BLAT [Kent, 2002], sacrifice some accuracy to gain dramatic performance improvements in return. Instead of exploring the reference genome entirely, heuristics *index* reference genomes in a way that enables for quick discovery of candidate alignment locations (CAL), which are then thoroughly examined by SW. Consequently, the accuracy of these heuristics is hinging on whether the correct alignment location in the reference genome is discovered among all CAL, such that the accuracy greatly varies depending on the implemented CAL-discovery method.

To obviate resulting inaccuracies, significant attempts were undertaken to revert to SW by taking advantage of massive parallel computing (MPC) on graphics processing units (GPUs) [Liu et al., 2009] or field-programmable gate arrays (FPGAs) (Convey Computer[1], 2012) that were found to outperform standard computing platforms by one to two orders of magnitude. Once more, rapidly evolving HTS techniques overwhelmed all these approaches, which were not only unable to cope with the dramatically increased sequence volumes emitted by HTS instruments, but also lacked technical requirements, such as full *gapped* SW alignment, support for paired-end reads or color-space encoded sequences. To keep pace, sequence alignment models were, again, rigorously redesigned to exploit two key technologies, namely (1) a *Suffix Array* (SA) [Manber and Myers, 1993] that represents a reference genome and permits extremely time-efficient discovery of CAL and (2) the *Burrows-Wheeler Transformation* [Burrows et al., 1994] to compress the SA that would otherwise consume 16 times the space the original representation of the reference genome consumes. However, derived accelerations didn't come cheap as these next generation read mapping algorithms typically implement even coarser CAL-discovery heuristics, such as the *maximal exact match* (MEM) approach [Khan et al., 2009]. Consequently, modern read mapper implementations are routinely subjected to critical benchmarking to conclude on appropriate balancing between execution performance and according accuracy.

CLC bio offers high-performance read mapping solutions as part of *CLC Genomics Workbench*, *CLC Genomics Server* and *CLC Assembly Cell*. Workbench and Server products integrate powerful bioinformatic tools under a unified graphical user interface. In contrast, *CLC Assembly Cell* is a high-performance computing product line dedicated to users versed in operating software at the command line level. It is the purpose of both product lines to solve complex bioinformatic tasks, such as *de-novo* and *reference* assemblies, the latter nowadays better known as *Read Mapping*. In 2012, *CLC bio* introduced a read mapping algorithm that implemented the MEM approach, and the 2012 issue of this white paper concluded that this read mapper was superior to its predecessor in all disciplines, but consumed significantly more memory. This drawback was removed in the current *memory-efficient* read mapper version. This white paper is dedicated to a comparison of both the previous and the memory-efficient read mapper versions along with some of the most popular open-source read mappers, *BWA*, *Bowtie2* and *SMALT* (see section 8.4). Overall, we distinguish three benchmarks:

1. **Performance Benchmark of** *CLC Genomics Workbench/Server*. Mapping reads to a target genome is considered the very first step to make sense out of resequencing data. However, sorting raw mapping results by chromosomal mapping coordinates is an obligatory post-processing step to enable visualization or further downstream analysis, such as variation calling. This benchmark assesses the overall user experience of mapping reads to a reference genome by measuring and comparing the processing times spent on the entire end-to-end read mapping process, and during individual sub-processes.

[1] http://www.conveycomputer.com/Resources/Convey_Announces_Record_Breaking_Smith_Waterman_Acceleration.pdf

2. **Computational Resource Requirements of** *CLC Assembly Cell*. Running software generally occupies computer memory and consumes processing time. These two metrics are referred to as computational expenses and read mapping is considered computationally expensive. To evaluate computational requirements necessary to achieve performances measured in the first benchmark, this second benchmark assesses resource consumption by real-time monitoring read mappers during execution.

3. **Accuracy Benchmark of** *CLC Assembly Cell*. Smith-Waterman based sequence aligners are considered to always discover optimal alignment results, while modern read mapping implementations follow heuristic approaches that trade some accuracy for performance. To value read mapping performances, this benchmark assesses and compares the losses of accuracy by comparing heuristic mapping results to optimal results obtained from a purely Smith-Waterman based read mapper.

Each of the three benchmarks were exercised using four different human data sets (see section 8.5) from four different sequencing platforms and were run on identical hardware (see section 8.6) to create thorough and informative benchmarks.

Platform	Reads	Protocol	Mean length
Illumina Genome Analyzer II	1,339,740,542	paired-end	101
Roche 454-Titanium	2,801,862	single-end	569
Life Technologies Ion Torrent	11,660,934	single-end	251
PacBioRS	1,702,801	single-end	555

Table 1: Overview of sample data sets. (see also section 8.5)

3 Performance Benchmark of *CLC Genomics Workbench/Server*

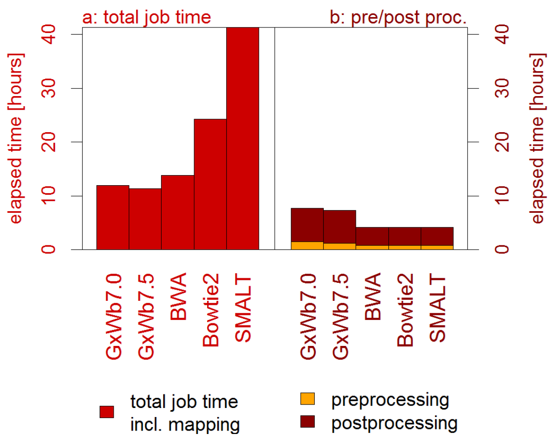
Among a broad range of bioinformatic tasks, mapping sequence reads to reference genomes doubtlessly plays a central role in *CLC Genomics Workbench/Server* products. Nonetheless, read mapping is typically just the first milestone along the way of digesting actual insight from sequenced cancer samples, family members or even a thousand individuals. In fact, variation calling upon high throughput sequencing (HTS) data has gained intense interest during the past decade and is currently considered the second milestone in many resequencing pipelines. To identify differences between samples, variation callers first compare individual samples separately to the reference genome and then intersect, subtract or otherwise compare resulting variation calls to discover inter-sample variations. For technical reasons, all variation callers require sequence mapping data to be sorted by reference genome coordinates. Since HTS instruments output sequencing reads in arbitrary order and read mappers maintain this order while deriving read mappings, sorting mapping results is obligatory. Sequence reads and read mapping data often display huge data volumes, such that sorting these data entities renders a computationally expensive operation, substantially contributing to the overall performance experienced by the user.

Another significant contribution to overall performance is displayed by data transfer operations. In many cases, HTS data is stored in dedicated storage facilities and have to be transferred to compute facilities on demand. Shoveling hundreds of gigabytes of sequencing data, as output by *Illumina HiSeq* instruments within just a single run, to a compute node that executes the actual read mapping, and analogously returning resulting mapping data to the storage facility, can easily contribute hours.

In fact, mapping sequence reads to a reference genome involves a three stage pipeline. In detail these are:

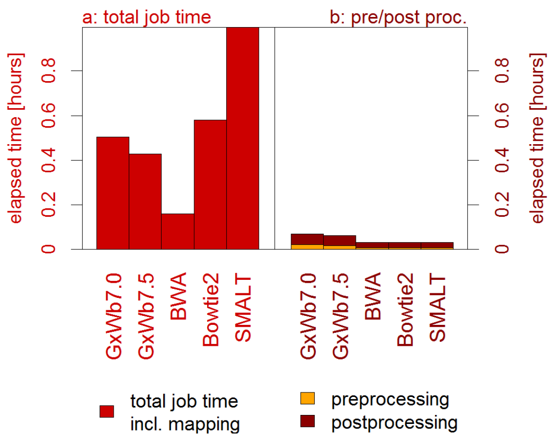
1. **pre-processing**
   (a) Raw sequence and reference genome data is copied to the compute nodes' local storage system. This is typically necessary, because high volume HTS data is mostly stored in storage units maintained by compute facilities. Also, in client/server setups, input data has to be transferred to the local storage of the compute nodes carrying out the read mapping.
   (b) The reference genome is then indexed. CLC mappers recreate the reference index for every read mapping run to enable high flexibility, e.g when using de-novo assemblies as reference genome. Building the index takes approx. 7 minutes for the full human genome (roughly 15 mins. on a standard laptop with a quad-core CPU). Identical Indexes, once created, are locally cached (on the grid-worker storage) to enforce maximum throughput for batching of small samples, such as amplicon data sets, in large reference genomes.
   (c) In the case of paired-end sequencing data, pre-processing also includes the automated assessment of the fragment size distribution by pre-mapping a representative subset of the original read pairs.
2. **read-mapping** A read mapper aligns input reads individually (or in pairs) to the reference genome and reports one or more mapping result per read (or read-pair) for downstream processing.
3. **post-processing**
   (a) The entirety of all derived mapping results are sorted by reference alignment coordinates to enable efficient visualization and to meet requirements of downstream analysis tools.
   (b) A mapping report is generated to communicate typical read mapping statistics, such as amounts and lengths of reads and references, but also details about reference genome coverage, zero-coverage regions, range of observed alignment identities, amounts of alternative mapping locations per read (mapping specificity), exact fragment size distributions in paired-end data sets, and so on.
   (c) Corresponding to step 1a, all result data is transferred to the storage unit.

Since the entirety of all three stages should be seen as one inseparable operation, all versions of *CLC Genomics Workbench/Server* streamline individual steps within a single integrated workflow. Overall runtimes of this *read mapping workflow* were measured for *CLC Genomics Workbench 7.0*, *CLC Genomics Workbench 7.5*, *BWA*, *Bowtie2* and *SMALT*. Each read mapper pipeline was benchmarked upon the four sample data sets (see section 8.5) and runtimes were compiled into figures 1, 2, 3 and 4. As expected, benchmarks of the *read mapping workflows* implemented in *CLC Genomics Workbench 7.0* (*GxWb7.0*) and *CLC Genomics Workbench 7.5* (*GxWb7.5*) show no significant difference, since the only anticipated difference is reduced main-memory consumption (see section 6).



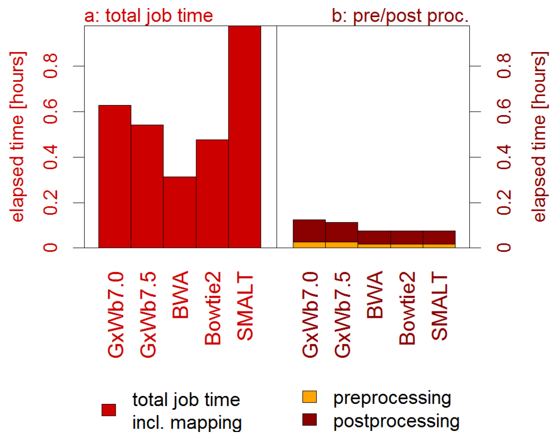
Mapper	total	preprocess	mapping	postprocess
GxWb7.0	11:55:47	1:31:10	4:13:49	6:10:48
GxWb7.5	11:20:35	1:16:41	3:58:43	6:05:11
BWA	13:52:24	0:50:28	9:42:37	3:19:19
Bowtie2	24:13:42	0:50:28	20:03:55	3:19:19
SMALT	41:16:29	0:50:28	37:06:42	3:19:19

Figure 1: Runtime measurements for the *Illumina* sample. The left subplot (a) indicates the total processing time, whereas the right subplot (b) shows only runtimes of pre- and post-processing, explicitly excluding the mapping runtime itself. Tabled runtimes are given in HH:MM:SS. Timings were individually measured for *CLC Genomics Workbench 7.0* (*GxWb7.0*), *CLC Genomics Workbench 7.5* (*GxWb7.5*) and for the open-source mappers *BWA*, *Bowtie2* and *SMALT*.



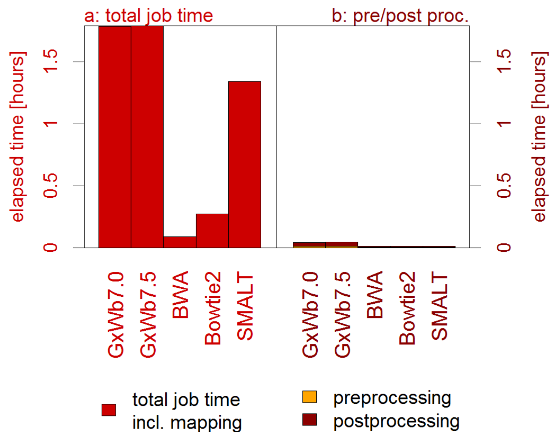
Mapper	total	preprocess	mapping	postprocess
GxWb7.0	0:30:19	0:01:20	0:26:03	0:02:56
GxWb7.5	0:25:44	0:01:02	0:21:56	0:02:46
BWA	0:09:40	0:00:33	0:07:44	0:01:23
Bowtie2	0:34:47	0:00:33	0:32:51	0:01:23
SMALT	0:59:42	0:00:33	0:57:46	0:01:23

Figure 2: Runtime measurements for the *454-Titanium* sample.



Mapper	total	preprocess	mapping	postprocess
GxWb7.0	0:37:43	0:01:34	0:30:15	0:05:54
GxWb7.5	0:32:28	0:01:38	0:25:44	0:05:06
BWA	0:18:48	0:01:00	0:14:16	0:03:32
Bowtie2	0:28:36	0:01:00	0:24:04	0:03:32
SMALT	0:58:35	0:01:00	0:54:03	0:03:32

Figure 3: Runtime measurements for the *Ion Torrent* sample.



Mapper	total	preprocess	mapping	postprocess
GxWb7.0	1:47:00	0:00:52	1:44:12	0:01:56
GxWb7.5	1:47:16	0:00:54	1:44:27	0:01:55
BWA	0:05:36	0:00:20	0:04:42	0:00:34
Bowtie2	0:16:34	0:00:20	0:15:40	0:00:34
SMALT	1:20:21	0:00:20	1:19:27	0:00:34

Figure 4: Runtime measurements for the *PacBioRS* sample.

Mapping pipelines based on open-source read mappers *BWA*, *Bowtie2* and *SMALT* lack the convenience of *CLC Genomics Workbench/Server* products, since none of them performs any of the aforementioned pre- or post-processing steps implicitly. Therefore transferal of read data and sorting, as well as storing of mapping data was carried out manually. Reference indexes were precreated (prior to benchmarking) and mapping report generation was entirely skipped. Still, *CLC Genomics Workbench 7.5* performs by far the quickest for the *Illumina* paired-end data set comprising 1.34 billion paired-end reads - an amount that roughly corresponds to what recent *Illumina HiSeq* instruments emit in few days (*Illumina*, [2], 2012). *CLC Genomics Workbench 7.5* performs competitively fast for the other samples, except for the *PacBioRS* sample, which naturally raises additional questions: what about computational resource requirements, and what about accuracy of mapping results?

4 Computational Resource Requirements of *CLC Assembly Cell*

Running software generally consumes processing time and occupies computer memory. These two metrics are referred to as the major computational expenses and aligning sequence reads to large reference genomes, such as the human or the murine, is considered computationally expensive. Overall runtimes of the three-stage mapping workflow, yielding immediately useable mapping data, were evaluated in section 3 and it was shown that the *read mapping* itself accounts for a major runtime portion of the described mapping workflow. It is therefore the purpose of this chapter to exactly assess overall computational requirements of running employed read mappers and to justify these by providing mapping statistics.

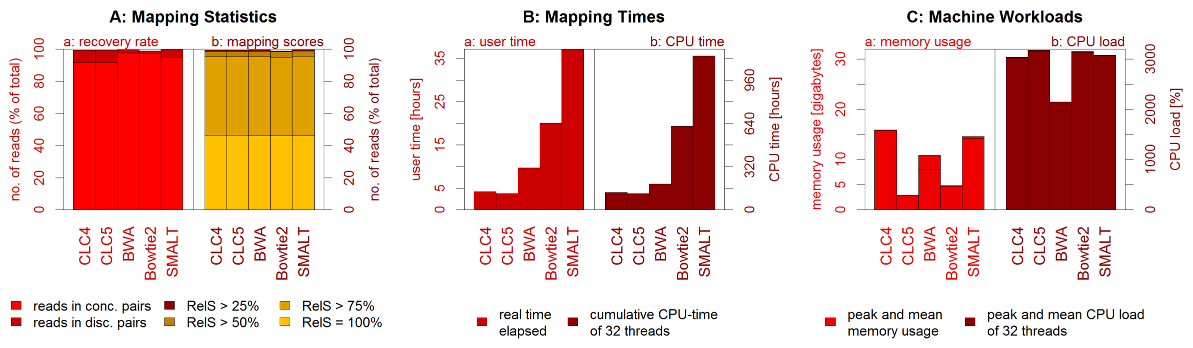
For the sake of clarity, it shall be mentioned that all *CLC Genomics Workbench/Server* products share identical core read mapping modules, such that mapping results and resource requirements measured here are identical for all products of the same generation. Table 2 provides an overview of *CLC* product lines and their major version numbers marking the transition from the old to the new read mapper.

Product line	**Latest version with old read mapper**	**Earliest version with new read mapper**
CLC Assembly Cell	4.2	5.0
CLC Genomics Server	6.0	6.5
CLC Genomics Workbench	7.0	7.5

Table 2: *CLC* product lines and employed default read mapper implementations. (see also section 8.4).

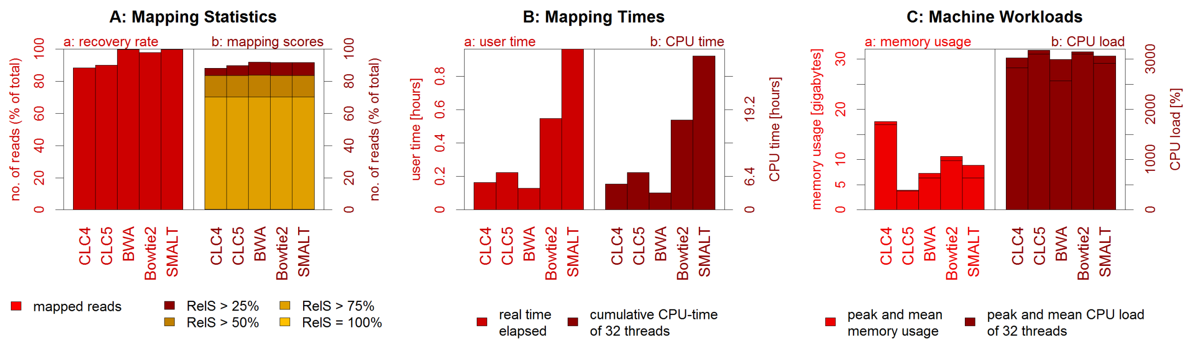
To compare computational requirements, resource consumptions were assessed for both the previous *CLC Assembly Cell 4.2* (*CLC4*) and the recently released *CLC Assembly Cell 5.0* (*CLC5*) along with the open-source read mappers *BWA*, *Bowtie2* and *SMALT*. As *CLC bio* provides general purpose software for various user groups, especially also addressing non-technicians, part of *CLC bio*'s efforts spent on software development is the *automagic* identification of optimal parameter settings where possible. As outlined in section 2, performance is negatively correlated to accuracy provided a fixed yet tunable implementation, i.e. the more a fixed implementation is tuned for accuracy, the slower it gets and vice versa. For simplicity, no attempts were undertaken to tune any of the mapper implementations in any of the benchmarks exercised, neither for speed nor for accuracy. Each read mapper was benchmarked upon the four sample data sets (see section 8.5) and benchmark results were summarized in figures 5, 6, 7 and 8.

[2] `http://www.illumina.com/systems/hiseq_systems.ilmn`



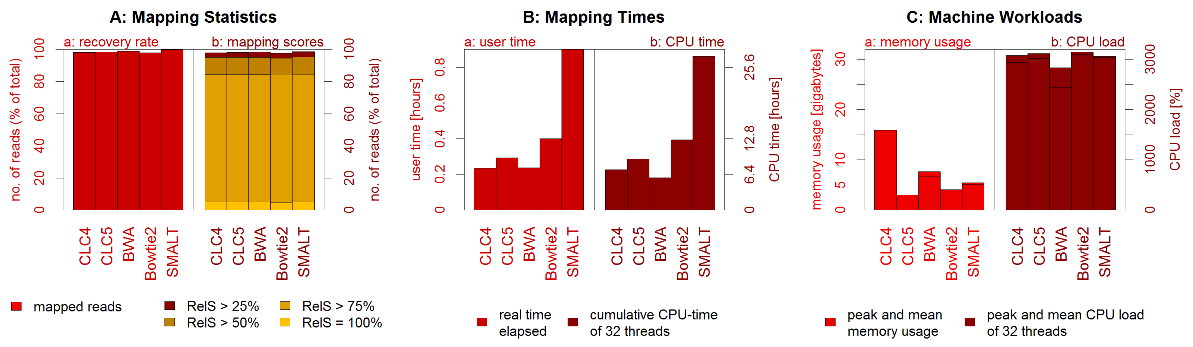
Mapper	unmapped	paired	unpaired	user time	cpu time	mem. load	cpu load
CLC4	0.66%	91.71%	7.64%	4:12:36	128:10:14	15.48	3,029%
CLC5	0.54%	91.79%	7.67%	3:45:29	119:06:50	2.88	3,167%
BWA	0.29%	97.77%	1.95%	9:42:37	189:57:14	7.85	1,964%
Bowtie2	1.19%	97.37%	1.44%	20:03:55	617:52:36	4.53	3,137%
SMALT	0.04%	94.96%	5.00%	37:06:42	1138:23:23	13.87	3,069%

Figure 5: Resource statistics for the *Illumina* sample (total reads: 1,339,740,542). Left plot (A) shows a mapping statistics summary: *recovery rate* reflects the percentage of reads of total that were successfully mapped as concordant pair or as broken pair; *mapping scores* give the percentage of reads that were mapped with a score higher than the given relative score threshold, where relative scores calculate as: *absolute score / read length in nucleotides*. Middle plot (B) summarizes runtime measurements: *user time* refers to the real time that was spent until the entire data set was processed; *CPU time* gives the total cumulative processor time consumed (table values in [H]H:MM:SS). Right plot (C) presents resource loads: *memory usage* (table values in Gigabyte) reflects the peak and (arithmetic) mean memory consumptions observed (table shows means only); *CPU load* indicates the peak and (arithmetic) mean processor loads in 0 - 3200 % according to 32 threads (table shows means only). Resource metrics were individually measured for *CLC Assembly Cell 4.2* (*CLC4*), *CLC Assembly Cell 5.0* (*CLC5*) and for the open-source mappers *BWA*, *Bowtie2* and *SMALT*.



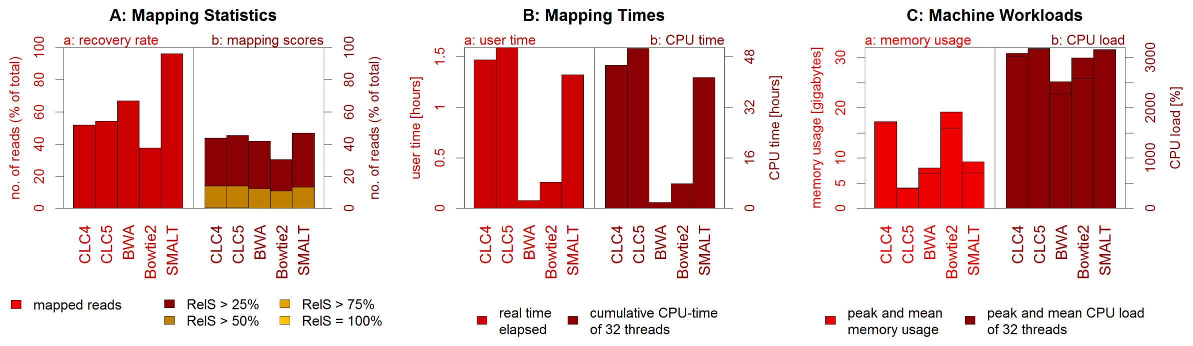
Mapper	unmapped	mapped	user time	cpu time	mem. load	cpu load
CLC4	11.52%	88.48%	0:09:54	4:59:37	16.61	2,832%
CLC5	9.83%	90.17%	0:13:29	7:08:19	3.69	3,106%
BWA	0.25%	99.75%	0:07:44	3:17:27	6.25	2,570%
Bowtie2	2.18%	97.82%	0:32:51	17:14:34	9.61	3,101%
SMALT	0.19%	99.81%	0:57:46	29:29:12	6.24	2,923%

Figure 6: Resource statistics for the *454-Titanium* sample (total reads: 2,801,862).



Mapper	unmapped	mapped	user time	cpu time	mem. load	cpu load
CLC4	1.87%	98.13%	0:14:06	7:14:42	15.36	2,939%
CLC5	1.57%	98.43%	0:17:39	9:10:03	2.94	3,019%
BWA	1.16%	98.84%	0:14:16	5:47:05	6.58	2,445%
Bowtie2	2.07%	97.93%	0:24:04	12:37:43	3.93	3,105%
SMALT	0.17%	99.83%	0:54:03	27:36:44	4.94	3,039%

Figure 7: Resource statistics for the *Ion Torrent* sample (total reads: 11,660,934).



Mapper	unmapped	mapped	user time	cpu time	mem. load	cpu load
CLC4	48.08%	51.92%	1:28:15	45:23:16	16.56	3,019%
CLC5	45.81%	54.19%	1:35:28	50:44:07	3.89	3,176%
BWA	33.20%	66.80%	0:04:42	1:53:57	6.71	2,281%
Bowtie2	62.47%	37.53%	0:15:40	07:49:03	15.59	2,587%
SMALT	3.64%	96.36%	1:19:27	41:30:38	6.91	3,128%

Figure 8: Resource statistics for the *PacBioRS* sample (total reads: 1,702,801).

As already indicated in section 3, mapping statistics generated here confirm that the memory-efficient read mapping module introduced in *CLC Assembly Cell 5.0* (*CLC5*) does not notably differ from the previous one in *CLC Assembly Cell 4.2* (*CLC4*). In fact, the entire benchmark shows almost identical results for the two versions except for the memory consumption that was reduced from approx. 16 Gb in *CLC4* to less than 4 Gb in *CLC5*. Where *CLC4* employed an uncompressed *Suffix Array* as underlying index structure for the reference genome, *CLC5* uses the *Burrows-Wheeler* transformation to build a FM-index [Ferragina and Manzini, 2000, Ferragina and Manzini, 2005] of the reference genome (see also section 6). While this seems to be a technical detail, the practical impact is significant; with *CLC5*, it is possible to create the index for a large reference genome, such as the human, in as little as 15 minutes on a standard laptop with no more than 4 processor cores and 8 Gb of RAM. Once the FM-index is created, it consumes even less, in the case of the human genome less than 3 Gb. Consequently, *CLC5* can be run on a standard laptop, which becomes increasingly attractive with growing amounts of cores made available in desktop processors. Note that the measured resource requirements apply analogously to all *CLC* products, since the core read mapping module is shared by *CLC Genomics Workbench 7.5*, *CLC Genomics Server 6.5* and *CLC Assembly Cell 5.0*, such that all *CLC* products take advantage of the reduced memory footprint in *CLC5*.

Comparing both *CLC* read mapping implementations to the open-source tools *BWA*, *Bowtie2*, and *SMALT* confirms the hugely superior performance of *CLC5* for the *Illumina* sample. Yet, *CLC5* runs notably longer for the *PacBioRS* data set. Here, specifically *BWA* is by far the fastest (see figure 8, subplot (B.a)) recovering more than 65% of all reads (see figure 8, subplot (A.a)). *Bowtie2* delivers solid accuracy and has become considerably faster since it left its beta-phase. In contrast, *SMALT* has a tendency towards long runtimes, peaking for the *Illumina* sample, but justifies the additional hours by reporting more than 90% of all *PacBioRS* reads mapped. Careful inspection of the mapping statistics uncover a few peculiarities: *CLC* read mappers report more broken paired reads (see figure 5, table column *unpaired*), which is due to a conscious decision as to which criteria two reads of a pair have to satisfy to be mapped as a pair as opposed to being mapped as a broken pair to two different reference positions. While assumptions resulting in this decision are clearly of contrary nature, it is ultimately up to downstream analysis tools to interpret mapping results, including information about broken pairs, in the context of biological relevance. Furthermore, *CLC* read mappers seem to report fewer alignments at very low relative scores, which is best observed in the *454-Titanium* and the *PacBioRS* samples, where *CLC5* reports notably fewer mapped reads than *SMALT* and *BWA* (see figures 6, and 8, see subplots (A.b)). In the case of imperfectly aligning reads, read mappers have to decide whether or not to report an alignment based on some minimum alignment quality thresholds, such as alignment identity (termed *similarity*) or minimum amount of read-nucleotides comprised by the alignment (termed *length-fraction*). This decision is again based on purely theoretical assumptions and might differ from read mapper to read mapper. For *CLC* read mappers these thresholds are more strict, which ultimately results in fewer reads reported to align at very low alignment scores. Note that amounts of reads aligned at relative scores higher than 50% do not notably differ.

To conclude, this benchmark shows in detail that *CLC5* performs identically to *CLC4* with regard to accuracy, but with a significantly reduced memory footprint. However, though not to withdraw, mapping statistics provided in this section can only give indications as to how accurate read mappers actually perform. In fact, it is almost irresponsible to conclude on accuracy based on overly simplifying metrics, such as amounts of mapped reads or nucleotides. Consequently, the ultimate question remains: if *CLC bio* is lightening fast, and runs on a laptop - what about true accuracy metrics?

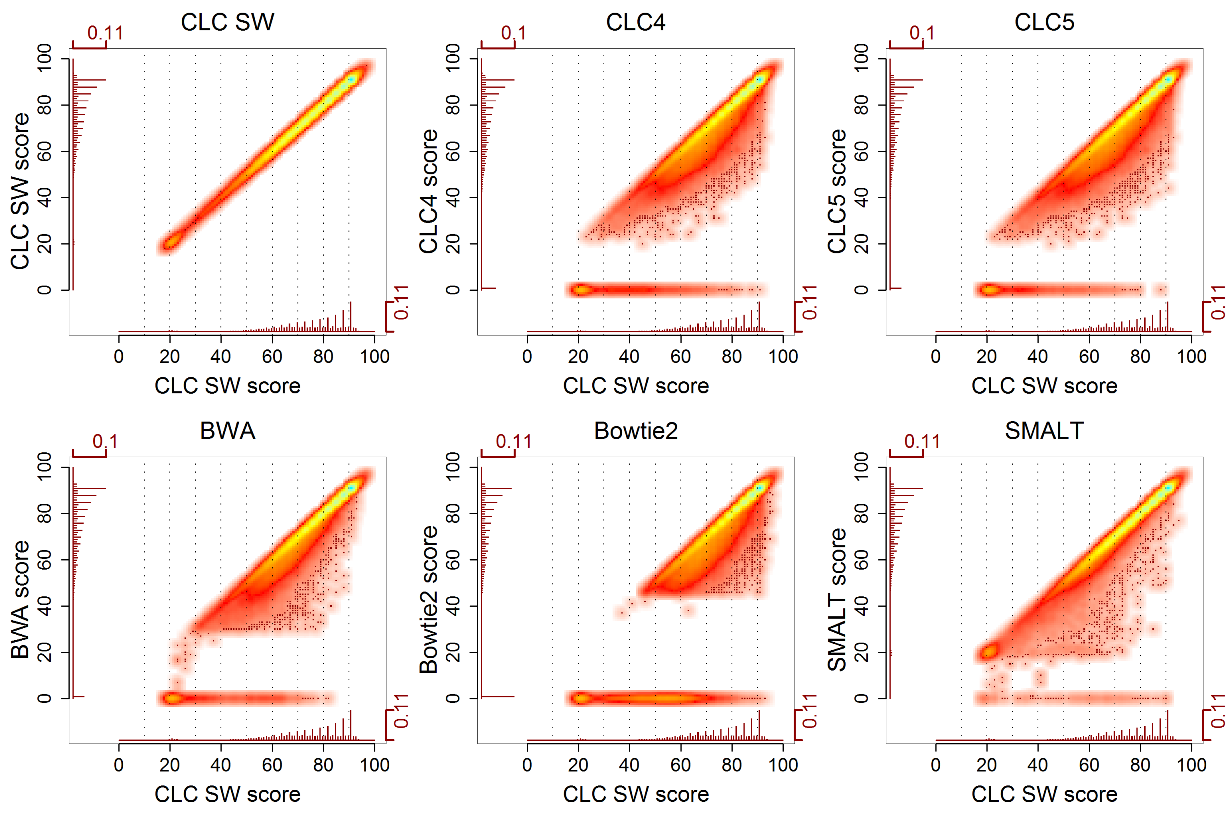
5 Accuracy Benchmark of *CLC Assembly Cell*

Assessing read mapping accuracy is required, because modern read mappers avoid performing a full Smith-Waterman (SW) alignment that, on the one hand, always generates perfect alignments, is, on the other hand, computationally too expensive to map billions of reads to large reference genomes. However, read mappers do not fully avoid SW, but utilize index data structures representing the reference genome to predetermine candidate alignment locations (CAL) that are eventually fully explored by SW. This results in a dramatic speed up, because SW needs to be performed only for the previously identified CAL, i.e. narrow regions, instead of the entire reference genome. Unfortunately, success of this approach is only guaranteed as long as sequence reads align perfectly to the reference genome. As soon as sequence reads contain sequencing errors or true variations, mapping accuracy is affected, because the heuristic step of identifying CAL is no longer guaranteed to discover the true alignment location among the false positive CAL. Thus, the accuracy of a read mapper strongly relies on the capability of identifying the true alignment location among all CAL, especially in the presence of sequencing errors or variations. Importantly, sequences that do in fact contain true variations are precisely those of most interest in many resequencing scenarios, such as research of genetic diseases. It is therefore absolutely essential that read mappers implement a reasonable CAL heuristic.

The previous sections focused on assessing performance metrics and computational requirements. It was shown that the read mapper implementation recently released as part of *CLC Assembly Cell 5.0* (*CLC5*), *CLC Genomics Workbench 7.5* and *CLC Genomics Server 6.5* has a significantly smaller memory footprint than its predecessor *CLC4*, but performs as fast. Some accuracy indicators were also presented, but it is still to be shown that accuracy hasn't suffered. Thus, the benchmarks presented in this section conclusively measure mapping accuracy.

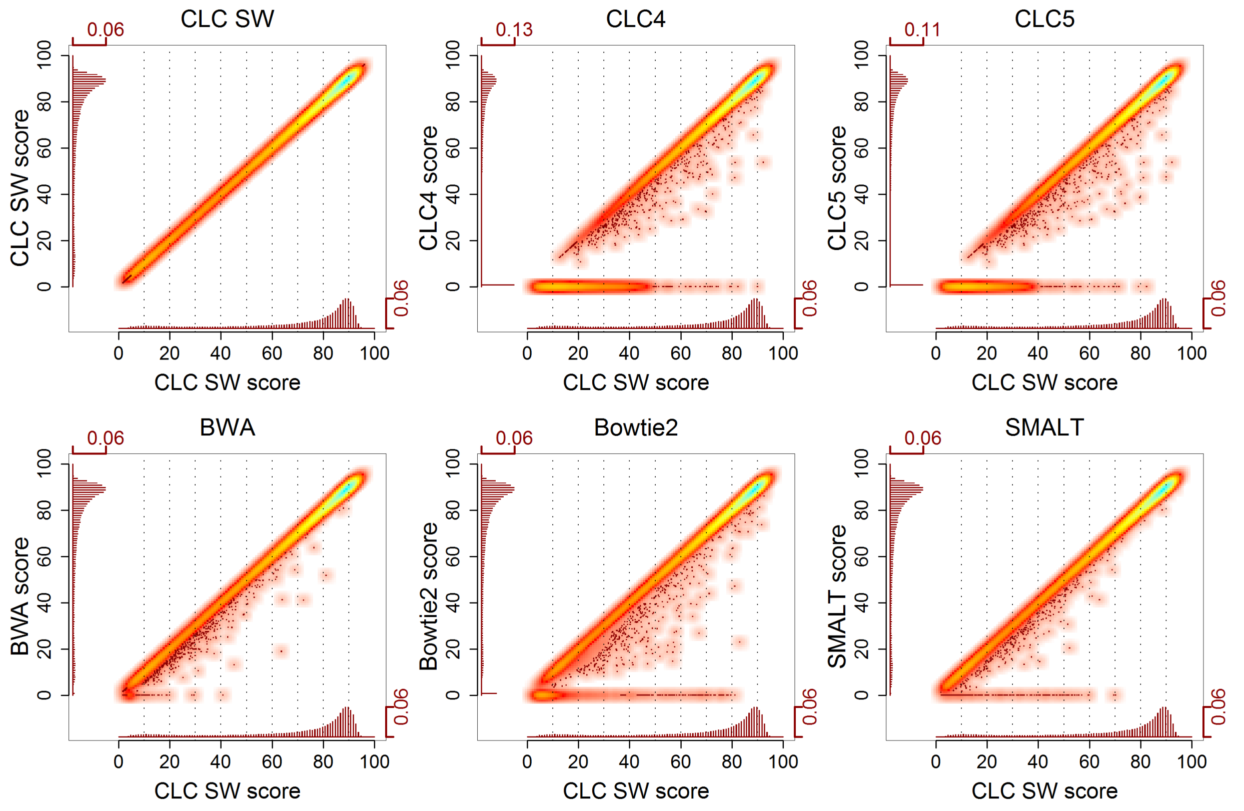
Assessing accuracy is, however, difficult in itself: various commercial and academic benchmarks have based their accuracy assessments on simulated reads that were extracted from a reference genome, then slightly altered according to estimated error modes of various sequencing platforms and finally mapped back to the reference genome. Accuracy was ultimately described as the proportion of reads that could be mapped back to the original reference genome position in spite of simulated sequencing errors. At least two major flaws of such an approach was recently discussed ([Holtgrewe et al., 2011]): first off, simulating reads by extracting small regions from a reference genome and introducing artificial errors requires exact knowledge of the error-modes produced by the targeted sequencing platform to generate representative error patterns; and second, even if generation of such data set succeeds, it is neither guaranteed that the original reference origin of a simulated read bearing simulated errors is still the best in terms of sequence similarity nor that it is the *only* alignment location, e.g. in case the simulated read was coincidentally taken from a repetitive element.

Accuracy assessments in this benchmark are thus based on a more sophisticated approach that was similarly described in the Holtgrewe manuscript: Four subsets of the original four data sets (see section 8.5) that only consist of sequence reads that *im*perfectly align to the reference genome were sampled. This was primarily done, because perfect sequence reads containing no errors at all are very often observed in *Illumina* data sets and to a lesser extend in *Ion Torrent*, *454-Titanium* and *PacBioRS* data. However, accuracy assessments using perfectly aligning sequences is inconclusive, since accuracy is a matter of identifying the true alignment location among the candidate alignment locations in presence of errors or variations. As opposed to simulated reads, the four resulting *complex* data subsets comprise genuine reads bearing platform-specific error profiles. *Golden-standard* mapping results were produced for these complex subsets by using a special read mapper implementation that skips identification of CAL and immediately performs full SW upon the entire reference genome instead. The heuristic read mappers then mapped these four complex data subsets, too, and mapping results for individual reads were compared to the *golden-standard*. Finally, accuracy was measured as proportion of reads that were aligned *optimally* by the heuristic mappers, where the alignment score was used as an underlying metric to judge whether or not an alignment was optimal, ultimately rendering a strong and comprehensible benchmark. *CLC4* and *CLC5* were used in this benchmark, especially to prove accuracy of *CLC5*, despite its reduced memory footprint. To summarize, the basic assumption of this accuracy benchmark is that SW always identifies the optimal alignment and that therefore no heuristic read mapper can ever discover an alignment that is more optimal in



Mapper	mapped	optimal	FDR	opt 50%	opt 60%	opt 70%	opt 80%	opt 90%
CLC SW	100.00%	100.00%	0.00%	8.02%	17.06%	24.35%	28.96%	15.38%
CLC4	95.57%	64.59%	32.42%	3.32%	7.83%	14.07%	23.89%	14.43%
CLC5	96.47%	64.94%	32.68%	3.32%	7.83%	14.07%	23.89%	14.43%
BWA	96.36%	72.03%	25.24%	3.19%	8.72%	16.95%	26.58%	15.15%
Bowtie2	88.96%	60.41%	32.09%	2.56%	6.42%	12.78%	23.83%	14.48%
SMALT	99.73%	83.93%	15.84%	4.73%	12.43%	21.20%	27.89%	15.11%

Figure 9: Accuracy statistics for the *Illumina* subsample (total reads: 100,000). Six subplots (A-F) present cross-comparisons of Smith-Waterman (SW) based mapping results and mapping results of five heuristic read mappers, where subplot (A) compares SW-based mapping results to themselves to provide reference shapes. Each subplot consists of a central scatterplot and two histograms nearby the axes. The histograms along the x-axes show percentages of reads mapped by SW at alignment scores indicated by the main x-axes. Y-scalings of these histograms are given at the right lower corners of each subplot. Similarly histograms along the y-axes show percentages of reads mapped by the actual read mappers at alignment scores indicated by the main y-axes. Y-scalings of those histograms are given at the upper left corners of the subplots. The central scatterplots correlate relative alignment scores for each individual read and thus contain 100,000 dots according to the total number of reads. To represent dots piling up at identical scatterplot-coordinates, a smoothed density profile was added. All plots are based on relative alignment scores, which are calculated as quotient of absolute alignment score divided by absolute read length in nucleotides. The table supplements the plots by denoting the number of mapped reads, the total number of optimal mappings and the resulting false discovery rate (FDR) (see section 8.3 for definitions). Similarly to FACS-gating, amounts of optimal alignments were assessed in 10%-intervals (see dotted verticals in plots) of SW-scores. All values given in the table are percent of total reads. Accuracy metrics were individually measured for *CLC Assembly Cell 4.2* (*CLC4*), *CLC Assembly Cell 5.0* (*CLC5*) and for the open-source mappers *BWA*, *Bowtie2* and *SMALT*.

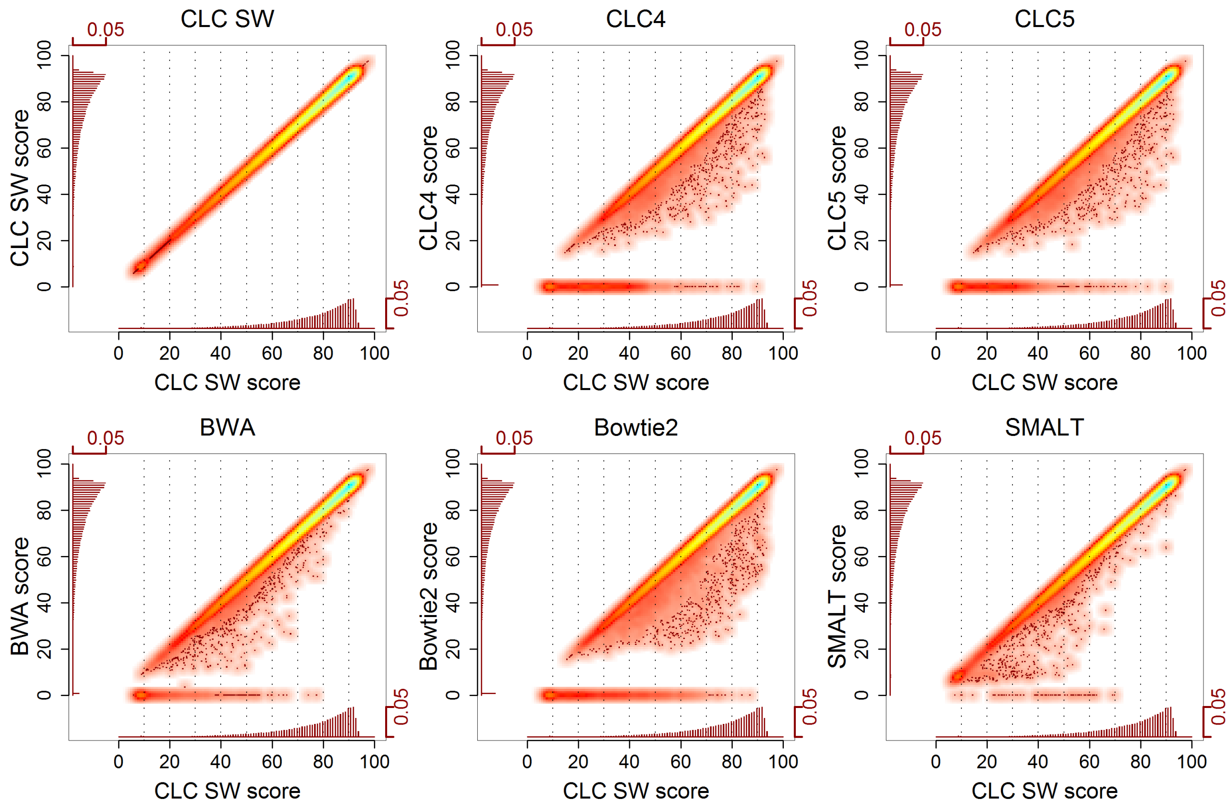


Mapper	mapped	optimal	FDR	opt 50%	opt 60%	opt 70%	opt 80%	opt 90%
CLC SW	100.00%	100.00%	0.00%	5.36%	7.92%	14.29%	40.45%	13.81%
CLC4	87.04%	86.13%	1.04%	5.21%	7.79%	14.19%	40.34%	13.78%
CLC5	88.92%	87.94%	1.11%	5.21%	7.79%	14.19%	40.34%	13.78%
BWA	99.64%	96.09%	3.57%	5.16%	7.71%	14.12%	40.33%	13.80%
Bowtie2	97.16%	91.82%	5.49%	4.91%	7.38%	13.69%	39.82%	13.70%
SMALT	99.80%	91.99%	7.83%	4.64%	7.06%	13.34%	39.46%	13.64%

Figure 10: Accuracy statistics for the *454-Titanium* subsample (total reads: 100,000).

the sense that it scores higher. However, read mappers can indeed report alignments scoring lower than the SW-based alignment, which certainly has to be interpreted as a failed attempt to discover the optimal alignment.

Comparing mappers to the golden-standard results suggests at least three populations: amounts of optimal alignments, amounts of suboptimal alignments and amounts of unmapped reads. Certainly, the higher the amount of optimally aligned reads, the better. In contrast, the higher the amounts of suboptimal alignments, the worse. Unmapped reads, however, must be ignored at least to some extend, because read mappers may decide to not report a specific alignment corresponding to some minimum alignment criteria. Still, it is clearly a mistake to not report an alignment for a read that is proven by SW to align with a close-to-maximum score. In general, the higher the reported SW-score, the more serious is the mistake of a mapper to report a suboptimal alignment or even none at all, indicating that the read mapper suppresses SW-exploration of too many CAL, has a bad CAL prioritization or does not discover promising CAL, at all. Hence, it does not matter, whether a read is in fact unmapped or suboptimally mapped, since both cases indicate that the CAL heuristic failed. Figures 9, 10, 11 and 12 are designed to easily spot these weaknesses. The higher the SW-based alignment score for an individual read, the more to the



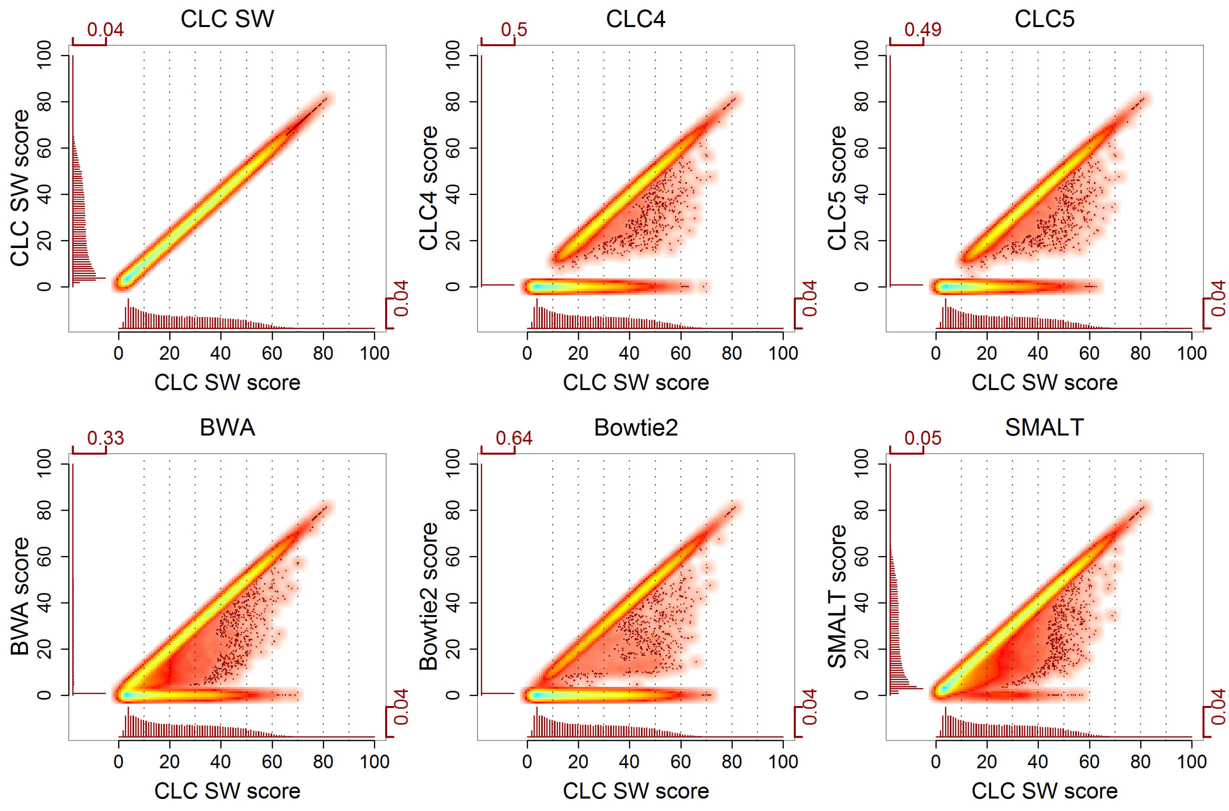
Mapper	mapped	optimal	FDR	opt 50%	opt 60%	opt 70%	opt 80%	opt 90%
CLC SW	100.00%	100.00%	0.00%	7.05%	12.17%	20.77%	36.99%	14.73%
CLC4	97.33%	93.97%	3.46%	6.30%	11.46%	20.29%	36.65%	14.66%
CLC5	97.97%	94.51%	3.53%	6.30%	11.46%	20.29%	36.65%	14.66%
BWA	98.92%	96.44%	2.51%	6.49%	11.80%	20.59%	36.91%	14.71%
Bowtie2	97.73%	93.84%	3.98%	6.10%	11.29%	20.28%	36.51%	14.61%
SMALT	99.93%	96.93%	3.01%	6.43%	11.72%	20.63%	36.92%	14.72%

Figure 11: Accuracy statistics for the *Ion Torrent* subsample (total reads: 100,000).

right the according dot appears in the scattered plot. The lower the mapper-based score reported for that read, the more to the bottom the dot is located, and in the extreme case of no reported alignment, the dot appears exactly on the x-axis. In contrast, dots corresponding to optimally aligned reads are located on the first diagonal as perfectly resembled in subplots (A). Similarly to FACS-gating, amounts of optimal and suboptimal alignments were assessed in 10%-gates of SW-scores. To recapitulate, read mappers have to be considered the more inaccurate, the fewer optimally aligned reads they report, most importantly in the right-most gates.

Accuracy results yielded upon the *Illumina* sample show that *CLC4* and *CLC5* perform virtually identical and differ only marginally from the open-source mappers. *BWA* benefits significantly from its new *MEM*-implementation, allowing short *Illumina* reads to be mapped in *local* mode, too. *Bowtie2* retained its accuracy despite the acceleration it gained since leaving beta-stage, and *SMALT* impresses with a very low FDR.

The *454-Titanium* and *Ion Torrent* based samples were very accurately mapped by all mappers. Had it not been for the strong minimum alignment criteria that force the *CLC* mappers to discard more reads below the 50% score threshold than the other read mappers, there were virtually no



Mapper	mapped	optimal	FDR	opt 20%	opt 30%	opt 40%	opt 50%	opt 60%
CLC SW	100.00%	100.00%	0.00%	16.90%	15.84%	13.12%	7.52%	1.83%
CLC4	49.74%	46.47%	6.57%	9.35%	12.47%	11.95%	7.18%	1.78%
CLC5	51.43%	47.82%	7.03%	10.04%	12.79%	12.01%	7.18%	1.78%
BWA	67.34%	59.60%	11.50%	11.33%	12.31%	11.81%	7.17%	1.78%
Bowtie2	35.66%	33.71%	5.47%	5.52%	7.44%	8.91%	6.26%	1.66%
SMALT	98.78%	73.57%	25.52%	13.25%	13.67%	12.33%	7.33%	1.80%

Figure 12: Accuracy statistics for the *PacBioRS* subsample (total reads: 100,000).

differences (see figures 10 and 11).

The *PacBioRS* sample flips the scenery (see figure 12). *SMALT* must be awarded the leading position in terms of accuracy, despite it is the slowest implementation overall. *BWA* achieves the second best result being the fasted at the same time. *CLC4* and *CLC5* perform equally well, but owing to comparatively strong minimum alignment criteria, worse than the aforementioned. Similarly, but to a greater extend, *Bowtie2* trails behind with the standard parameters applied here.

To summarize, *CLC5* generates as accurate results as its predecessor *CLC4* throughout all sample types and benchmark categories. Even more, all benchmarks indicate that *CLC5* remains among the best read mappers on the market, especially considering that *CLC5* automagically adapts to arbitrary error-modes of any kind with out user guidance and produces consistently reliable results across all data types.

6 Algorithmic Details

The implementations of the read mappers underlying *CLC4* and *CLC5* do not vary greatly and still *CLC5* is a major upgrade. Thus, this chapter discusses some of the major algorithm details to provide a basis for a better understanding of peculiarities observed.

CLC Assembly Cell 4.2 (*CLC4*) is based on an uncompressed Suffix-Array (SA) representing the entire reference genome in a single data structure. The algorithm iterates over input reads and maps each read individually by applying the following procedure: initially, the longest stretches of matching base pairs between reference genome and read are determined by considering each base position of the read as the start position of a seed candidate. End-positions of seeds are determined by elongating seed candidates as long as they are identical to the reference. This approach is well examined and known as *maximal exact match* (MEM) [Khan et al., 2009]. Resulting seeds are maintained in a list of seeds that might overlap each other, but may not include each other. After all possible seeds for a read are found, the list of resulting seeds is prioritized by descending length and read offset. Finally, a maximum of 100 seeds is examined in detail using a banded Smith-Waterman algorithm. Memory consumption of *CLC4* is bounded from below by $5 * N$, where N equals the size of the reference genome, such that the SA of the human genome (approx. length: 3 GBases) consumes 15 GBytes of main memory. Whilst such a requirement might appear inconsiderate, it allows for very fast index creation, which is a defining goal when maintaining flexibility with respect to hard-masking or otherwise altering the reference genome, for example when mapping reads back to de-novo assembly with thousands of contigs. Reducing the memory footprint of *CLC4* can be achieved by Burrows-Wheeler transforming the SA and thus creating a FM-Index [Ferragina and Manzini, 2000, Ferragina and Manzini, 2005] of the reference genome. This transformation is computationally expensive and is very hard to parallelize. For comparison, constructing the SA takes roughly 10 minutes, constructing the FM-index takes both *BWA* and *Bowtie2* more than 90 minutes.

CLC Assembly Cell 5.0 (*CLC5*) uses the exact same alignment strategy as *CLC4*. Neither seeding nor subsequent extension phase were changed significantly, rendering resulting mappings essentially identical. However, rather than using a memory-intensive SA, *CLC5* in fact employs a Burrows-Wheeler transformation of the reference genome. The previously described Burrows-Wheeler algorithm [Burrows et al., 1994] is based on the transformation of the concatenation of all chromosomes of a reference genome as one long string. In contrast, *CLC5* parallelizes the transformation of individual chromosomes and subsequently merges the transformations. On a large server with 24 processor cores all human chromosomes are transformed in parallel and then merged into a single index. On smaller laptops the number of chromosomes transformed in parallel is initially determined by the number of processor cores, and then additionally limited by the amount of main memory available. If a laptop has only enough memory to transform two chromosomes in parallel, then only two transformations will be carried out at a time, i.e. only two processor cores will be used. This algorithm allows *CLC5* to construct the FM-index for the human genome in roughly 5 minutes on a server (with 16 core and 32Gb of memory) and in 15 minutes on a laptop (4 cores and 8Gb of memory).

7 Conclusions

This white paper compares the previous to the improved read mapping solution underlying all *CLC bio* products, and correlates results to solutions provided by the academic community.

Initial benchmarks showed a significant reduction of main memory (RAM) required in *CLC Genomics Workbench 7.5* and *CLC Genomics Server 6.5* during mapping of sequence reads to large reference genomes, such as the human or the murine. Retained convenience and ease of use were demonstrated in many regards: a single integrated workflow takes all necessary steps to provide read mapping results directly re-usable in downstream analyses, and implicitly generates mapping reports. These reports provide detailed information about the input data sets, and describe in great detail the metric traits of the yielded mapping results.

CLC Assembly Cell comprises command line versions of the read mapping solutions underlying *CLC Genomics Workbench/Server*. Computational resource consumptions were assessed by real-time monitoring individual read mappers during execution. Subsequent evaluations demonstrated that the new read mapper scales even better with multi-processor machines. Required main memory was shown to be greatly reduced to an amount that is typically available in standard laptops. This enables read mapping, as implemented in *CLC Genomics Workbench 7.5* and *CLC Assembly Cell 5.0*, to be performed on small machines, specifically for small samples, where processor power is of secondary interest.

Finally, strong accuracy metrics were developed and *CLC Assembly Cell* was critically benchmarked upon these. It was shown that mapping accuracy of *CLC5* is, despite its superior performance and reduced resource consumption, neck and neck with its predecessor *CLC4* for all sequencing technologies. To this end, *CLC5* can serve as a drop-in replacement for *CLC4*, even more as command line interfaces are identical. *CLC Genomics Workbench 7.5* and *CLC Genomics Server 6.5* take implicit advantage of the improvements as *CLC5* underlies all *CLC bio* products.

To ultimately conclude: accuracy of biologically relevant conclusions derived from sequencing data strongly hinge on accurate read mapping being the initial step in resequencing pipelines. This white paper conclusively demonstrated that *CLC Genomics Workbench 7.5*, *CLC Genomics Server 6.5*, and *CLC Assembly Cell 5.0* have the technological capabilities to handle sequence reads from any of the current sequencing platforms and in conjunction with large reference genomes, such as the human. All *CLC bio* products are available for all major platforms, including *Linux/Unix*, *OS X*, and especially *Microsoft Windows®* both x86 and x64.

8 Materials & Methods

8.1 Performance Benchmark of *CLC Genomics Workbench/Server*

Runtimes were measured for *CLC Genomics Workbench 7.0*, *CLC Genomics Workbench 7.5* as well as for the open-source read mappers *BWA*, *Bowtie2* and *SMALT*. Since the latter do not implicitly undertake any of the described pre- or post-processing steps, copying of data sets was carried out using *bash* commands. SAMtools v0.1.19 [Li et al., 2009] were employed to sort and index raw mapping results. Since this step is the same for all employed open-source mappers, only postprocessing Bowtie2 results was benchmarked and results were identically included in benchmarks of all three open-source mappers, such that timings tabled in figures 1, 2, 3 and 4 are congruent.

Prior to mapping reads against reference genomes, read mappers typically require generation of indexes of the reference sequences. *CLC* read mappers create these indexes on the fly, when the read mapping is executed. For *BWA*, *Bowtie2* and *SMALT* prebuilding of indexes is required, which was carried out upfront and was explicitly excluded from the runtime measurements in this benchmark.

BWA comes in three different flavors, though BWA authors recommend BWA-MEM for data sets used here.[3] BWA-MEM was therefore used for all data sets.

8.2 Computational Resource Requirements of *CLC Assembly Cell*

To measure typical computational expenses the four sample data sets were mapped against the human reference genome using *CLC Assembly Cell 4.2*, *CLC Assembly Cell 5.0* as well as the open-source read mappers *BWA*, *Bowtie2* and *SMALT*. Since read mapper processes are neither expected to finish quickly nor to change their memory requirements very dynamically, real-time process profiling using genuine profiling solutions, such as *Valgrind* or *OProfile*, was neither necessary nor desired, since these can cause significant computational overheads by themselves. Instead, repeated calls to the unix-command *ps* were used to assess computational costs of executing the read mappers. To automate monitoring and yield a profile of resource consumptions, a bash-based wrapper script was implemented that supports to invoke a read mapper and to monitor its resource consumptions by repetitively calling *ps*. Amongst others, the wrapper script assessed the following metrics every 10 seconds and wrote them to a log-file for later evaluation:

- elapsed user time (= real time) since the monitored process was started
- cumulative CPU time of the monitored process
- current cpu utilization of the monitored process
- non-swapped physical memory consumption of the monitored process

Besides computational expenses basic mapping statistics were assessed upon the results derived per sample and read mapper. Since comparison of mapping results is per se challenging, scoring schemes of employed read mappers were adjusted to resemble match (+1), mismatch (-2) and gap/deletion (-3) rewards. No further attempts were made to tune any of the mapper implementations in any of the benchmarks exercised, neither for speed nor for accuracy. In fact all mappers were benchmarked using standard parameter settings except for those to specify and describe input data. This certainly includes settings to indicate the correct encoding scheme for base quality scores and the fragment-sizes (0 to 1,000 Nt) for paired-end data.

8.3 Accuracy Benchmark of *CLC Assembly Cell*

Benchmarking accuracy using sequences that perfectly align to the reference genome is inconclusive, since accuracy is a matter of identifying the true alignment location among the candidate alignment locations in presence of errors or variations. Complex data subsets, i.e. data sets that only contain reads that do not trivially align to the reference genome, had thus to be created. To this end, original data sets were initially mapped using *CLC Assembly Cell 4.2* in global alignment mode, requiring minimum alignment criteria of 98% similarity (aka. identity) and 98%

[3] see FAQ section on `http://http://bio-bwa.sourceforge.net`

length-fraction (portion of the read that is covered by the alignment). Reads that were reported unmapped according to above minimum alignment criteria, were subsequently extracted from the original input files, if their length was greater than 50 Nt. Of those, 100,000 reads per original sample were saved to four separate FASTQ files, representing *complex data subsets* used in the accuracy benchmarks. Listing 7 outlines the described workflow.

To discover the optimal alignments for those complex sequence reads, a special read mapper implementation was used that employs a full, gapped Smith-Waterman (SW) algorithm and supports aligning reads to the entire human reference genome. This SW-implementation achieves a performance of around 650 GCUPS on a machine with 32 physical Intel-cores and thus supports to map each of the four complex data subsets within one to five days. The four subsets were then mapped using the actual read mappers and yielded alignment scores were per read compared to the respective SW-scores. To guarantee fair comparisons, scoring schemes of all mappers were adapted to resemble match (+1), mismatch (-2) and gap/deletion (-3) rewards. To exclude biasing alignment refinements and /or prioritization, base-quality scores were stripped from the FASTQ, in fact by converting them to FASTA files. Moreover, all samples were mapped in single-end mode, specifically the originally paired-end *Illumina* sample, too. Pairwise score comparisons were finally inspected using descriptive statistics. Any read reported with an alignment score

- of 0 was interpreted as an unmapped read and thus counted as a failed attempt to identify the optimal alignment, because SW always returns an alignment.
- lower than the according SW-score was interpreted as a sub-optimally aligned read and thus counted as a failed attempt to identity the optimal alignment, because SW is by definition always right.
- equal to the SW-score was interpreted as an optimal alignment, no matter where in the reference genome this alignment was located, because the term 'optimal' refers to the highest possible alignment score.
- higher than SW-score was interpreted as an super-optimal alignment, which certainly never happened, because SW always finds the highest score.

In formal terms of sensitivity and specificity the following definitions were applied:

- FN: unmapped reads, because SW always reports an alignment (false negatives).
- TN: always 0, because SW always reports an alignment for every read (true negatives).
- FP: reads mapped with a suboptimal score compared to SW (false positives).
- TP: reads mapped with an optimal score compared to SW (true positives).

Calculating a specificity as opposed to sensitivity is obviously not useful, because TN is always 0. Instead accuracy (ACC) and false discovery rate (FDR) are calculated:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} = \frac{TP}{total\,reads}$$

$$FDR = \frac{FP}{FP + TP} = \frac{TP}{P} = \frac{TP}{mapped\,reads}$$

8.4 Read Mappers

Statically compiled Linux-x64 binaries of the following read mappers were used throughout all analyses. Were binaries were not directly available for download, GCC 4.4.7 was used to build them on *CLC Genomics Machine* (see section 8.6).

Mapper	Version	Abbreviation	Reference / Notes
CLC Assembly Cell	4.22.107091	*CLC4*	also included in *CLC Genomics Workbench 7.0* and *CLC Genomics Server 6.0*
CLC Assembly Cell	5.00.105852	*CLC5*	also included in *CLC Genomics Workbench 7.5* and *CLC Genomics Server 6.5*
BWA	0.7.9	*BWA*	[Li and Durbin, 2009]
Bowtie2	2.2.2	*Bowtie2*	[Langmead and Salzberg, 2012]
SMALT	0.7.5	*SMALT*	*unpublished*[4]

Table 3: Overview of read mapper versions employed for benchmarking.

8.5 Data Sets

Publicly available sample data sets sequenced from human DNA were used for all evaluations and benchmarks. Read data were downloaded from the source webpages in raw FASTQ format and individual lanes/chips/cells were concatenated to ease handling throughout the benchmarks. Neither quality trimming nor any other data preprocessing was applied to any of the data sets.

platform	units	src	read count	read length	volume
Illumina Genome Analzer II	25 lanes	[5]	669,870,271	100/102 Nt	136 GB
Roche 454-Titanium	2 flow cells	[6][7]	2,801,862	569 Nt	1.6 GB
Life Technologies Ion Torrent	2 Ion318 chips	[8]	11,660,934	251 Nt	2.89 GB
PacBioRs	8 C1-SRMT cells	[9]	1,702,801	555 Nt	0.95 GB

Table 4: Sample data sets.

The following human reference genome was used to map reads:

Species	Homo Sapiens
Assembly version	GRCh37 (HG19)
Source	*UCSC*[10]
Chromosomes	chr1-chr22, chrX, chrY, chrM (butï¿$\frac{1}{2}$not chrUn)

Table 5: Reference genome specification.

[4] http://www.sanger.ac.uk/resources/software/smalt/
[5] http://www.ebi.ac.uk/ena/data/view/ERP000460
[6] http://www.ebi.ac.uk/ena/data/view/SRR003161
[7] http://www.ebi.ac.uk/ena/data/view/SRR003162
[8] http://ioncommunity.lifetechnologies.com/docs/DOC-2162
[9] http://www.smrtcommunity.com/Share/Datasets/HapMap-Broad
[10] http://hgdownload.cse.ucsc.edu/goldenPath/hg19/bigZips/chromFa.tar.gz

8.6 Benchmarking Hardware

All evaluations and benchmarks were executed on a dedicated *CLC Genomics Machine*.

processors	2x Intel E5-2650ï¿½@ 2.00 GHz
total physical cores	16
total logical cores	32
main memory	64 Gbyte
internal storage	LSI-RAID5 of SATA-disks (6x 3TB)
operating system	CentOS 6.5 amd64
kernel version	Linux 2.6.32.x86_64

Table 6: Hardware specifications of *CLC Genomics Machine*.

8.7 *bash* related

The following section provides code listings to document exact parameter settings used throughout the various benchmarks.

```
# BWA
bwa index -p hg19 -a bwtsw hg19.fa &> hg19.log
# Bowtie2
bowtie2-build-s -f hg19.fa hg19 &> hg19.log
# SMALT
smalt index -k 13 -s 6 hg19 hg19.fa &> hg19.log
```

Listing 1: Creating reference genome indexes for open-source mappers.

```
# single-end
clc_mapper --cpus 32 --references hg19.fa --reads in.fq --output out.cas
# paired-end
clc_mapper --cpus 32 --references hg19.fa --reads -p fb ss 1 1000 -i in_1.fq in_2.fq --output
    out.cas
```

Listing 2: Mapping reads using *CLC Assembly Cell 4.2*

```
# single-end
clc_mapper_beta --cpus 32 --references hg19.fa --reads in.fq --output out.cas
# paired-end
clc_mapper_beta --cpus 32 --references hg19.fa --reads -p fb ss 1 1000 -i in_1.fq in_2.fq --
    output out.cas
```

Listing 3: Mapping reads using *CLC Assembly Cell 5.0*

```
# single-end
bwa mem -t 32 -A 1 -B 2 -O 0 -E 3 -L 0 hg19 in.fq > out.sam
# paired-end
bwa mem -t 32 -A 1 -B 2 -O 0 -E 3 -L 0 hg19 in_1.fq in_2.fq > out.sam
```

Listing 4: Mapping short reads using *BWA*

```
# single-end
bowtie2-align-s --threads 32 --phred33-quals --local --ma 1 --mp 2 --rdg 0,3 --rfg 0,3 -x hg19 in.fq > out.sam
#paired-end
bowtie2-align-s --threads 32 --phred33-quals --local --ma 1 --mp 2 --rdg 0,3 --rfg 0,3 -x hg19 --minins 0 --maxins 1000 -1 in_1.fq -2 in_2.fq > out.sam
```

Listing 5: Mapping short reads using *Bowtie2*

```
# single-end (inverse grep to strip '#'-styled comments from stdout)
smalt map -n 32 -f sam -O -S match=1,subst=-2,gapopen=-3,gapext=-3 hg19 in.fq | grep -v '^#' > out.sam
# paired-end (inverse grep to strip '#'-styled comments from stdout)
smalt map -n 32 -f sam -O -S match=1,subst=-2,gapopen=-3,gapext=-3 -j 0 -i 1000 hg19 in_1.fq in_2.fq | grep -v '^#' > out.sam
```

Listing 6: Mapping short reads using *SMALT*

```
# map reads requiring 98\% similarity and 98\% length-fraction
clc_mapper_legacy -l 0.98 -s 0.98 -a global -d hg19.fa -q in.fq -o out.cas
# extract reads that failed to align according to above constraints
clc_unmapped_reads -a out.cas -o complex_input.fq - l 50
# of those, save 100000 reads
head complex_input.fq -n 400000 > complex_subset.fq
```

Listing 7: Creating a complex data subset using *CLC Assembly Cell 4.2*

```
clc_mapper_sw --cpus 80 -s 0.0 -d hg19.fa -q in.fa -o out.cas
```

Listing 8: Mapping reads using Smith-Waterman

References

[Altschul et al., 1990] Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990). Basic local alignment search tool. *J Mol Biol*, 215(3):403–410.

[Bosch and Grody, 2008] Bosch, J. R. and Grody, W. W. (2008). Keeping up with the next generation: massively parallel sequencing in clinical diagnostics. *J Mol Diagn*, 10(6):484–492.

[Burrows et al., 1994] Burrows, M., Wheeler, D. J., Burrows, M., and Wheeler, D. J. (1994). A block-sorting lossless data compression algorithm.

[Ferragina and Manzini, 2000] Ferragina, P. and Manzini, G. (2000). Opportunistic data structures with applications. In *Proc. 41st Annual Symp. Foundations of Computer Science*, pages 390–398.

[Ferragina and Manzini, 2005] Ferragina, P. and Manzini, G. (2005). Indexing compressed text. *J. ACM*, 52(4):552–581.

[Holtgrewe et al., 2011] Holtgrewe, M., Emde, A.-K., Weese, D., and Reinert, K. (2011). A novel and well-defined benchmarking method for second generation read mapping. *BMC Bioinformatics*, 12:210.

[Kent, 2002] Kent, W. J. (2002). BLAT—The BLAST-Like Alignment Tool. *Genome Research*, 12(4):656–664.

[Khan et al., 2009] Khan, Z., Bloom, J. S., Kruglyak, L., and Singh, M. (2009). A practical algorithm for finding maximal exact matches in large sequence datasets using sparse suffix arrays. *Bioinformatics*, 25(13):1609–1616.

[Langmead and Salzberg, 2012] Langmead, B. and Salzberg, S. L. (2012). Fast gapped-read alignment with bowtie 2. *Nat Methods*, 9(4):357–359.

[Li and Durbin, 2009] Li, H. and Durbin, R. (2009). Fast and accurate short read alignment with burrows-wheeler transform. *Bioinformatics*, 25(14):1754–1760.

[Li et al., 2009] Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., Marth, G., Abecasis, G., Durbin, R., and , . G. P. D. P. S. (2009). The sequence alignment/map format and samtools. *Bioinformatics*, 25(16):2078–2079.

[Liu et al., 2009] Liu, Y., Maskell, D. L., and Schmidt, B. (2009). Cudasw++: optimizing smith-waterman sequence database searches for cuda-enabled graphics processing units. *BMC Res Notes*, 2:73.

[Manber and Myers, 1993] Manber, U. and Myers, G. (1993). Suffix arrays: A new method for on-line string searches. *SIAM Journal on Computing*, 22(5):935–948.

[Maxam and Gilbert, 1977] Maxam, A. M. and Gilbert, W. (1977). A new method for sequencing dna. *Proc Natl Acad Sci U S A*, 74(2):560–564.

[Sanger et al., 1977] Sanger, F., Nicklen, S., and Coulson, A. R. (1977). Dna sequencing with chain-terminating inhibitors. *Proc Natl Acad Sci U S A*, 74(12):5463–5467.

[Service, 2006] Service, R. F. (2006). Gene sequencing. the race for the $1000 genome. *Science*, 311(5767):1544–1546.

[Smith and Waterman, 1981] Smith, T. F. and Waterman, M. S. (1981). Identification of common molecular subsequences. *J Mol Biol*, 147(1):195–197.