MapCaller – An integrated and efficient tool for short-read mapping and variant calling using high-throughput sequenced data

Hsin-Nan Lin and Wen-Lian Hsu*

Institute of Information Science, Academia Sinica, Taipei, Taiwan

*To whom correspondence should be addressed (hsu@iis.sinica.edu.tw)

Abstract

With the advance of next-generation sequencing (NGS) technologies, more and more medical and biological researches adopt NGS technologies to characterize the genetic variations between individuals. The identification of personal genome variants using NGS technology is a critical factor for the success of clinical genomics studies. It requires an accurate and consistent analysis procedure to distinguish functional or disease-associated variants from false discoveries due to sequencing errors or misalignments.

In this study, we integrate the algorithms for read mapping and variant calling to develop an efficient and versatile NGS analysis tool, called MapCaller. It not only maps every short read onto a reference genome, but it also detects single nucleotide variants, indels, inversions and translocations at the same time. We evaluate the performance of MapCaller with existing variant calling pipelines using three simulated datasets and four real datasets. The result shows that MapCaller can identify variants accurately. Moreover, MapCaller runs much faster than existing methods. We demonstrate MapCaller is both time and space efficient for NGS data analysis. MapCaller is available at https://github.com/hsinnan75/MapCaller.

1 Introduction

With the advance of next-generation sequencing (NGS) technologies, it is becoming affordable to support various applications of precision medicine in the near future(Ku and Roukos, 2013). More and more medical and biological researches adopt NGS technologies to characterize the genetic variations between individuals (Dong, et al., 2015; Kulkarni and Frommolt, 2017). Such genetic variations can be classified into three types: (1) single nucleotide variant (SNV, also referred to as SNP); (2) insertion and deletion (indel); and (3) structural variant (SV, including translocation, inversion, copy number variation and indels of size at least 50 bp).

The identification of genome variants is a critical factor for the success of clinical genomics studies(Hwang, et al., 2015). It requires an accurate and consistent analysis procedure to distinguish true variants from false discoveries. This procedure often involves the steps of short read alignment, alignment rearrangement, and variant calling. In each step, one or more tools are applied to generate desired output. For example, BWA(Li and Durbin, 2009), Bowtie (Langmead and Salzberg, 2012; Langmead, et al., 2009), GEM (Marco-Sola, et al., 2012), Subread (Liao, et al., 2013), HISAT/HISAT2 (Kim, et al., 2015), and KART (Lin and Hsu, 2017) are read aligners that can map NGS short reads onto a reference genome and generate their alignments. SAMtools (Li, et al., 2009) and Picard (Carver and Stubbs, 1997)provide various utilities for manipulating read alignments. For variant calling, the Genome Analysis Tool Kit (GATK) (DePristo, et al., 2011), Freebayes (Erik Garrison, 2012), Platypus (Rimmer, et al., 2014), VarScan (Koboldt, et al., 2009) and SAMtools are widely used.

Different combinations of those tools produce various analysis pipelines. Thus different variant calling pipelines may generate substantial disagreements of variant calls. Several studies (Hwang, et al., 2015; Yen, et al., 2017) have been conducted to confirm the disagreements. Besides, all existing variant calling pipelines are time and space consuming. They require read alignments are sorted and stored in desired format.

In this study, we present MapCaller, an efficient and versatile NGS analysis tool, by integrating the algorithms for read mapping and variant calling. For read mapping, we adopt a divide-and-conquer strategy to separate a read into regions with and without gapped alignment. With this strategy of read partitioning, SNVs, indels and breakpoints can be identified efficiently. For variant calling, MapCaller maintains a position frequency matrix to keep track of every nucleobase's frequency at each position of the reference genome while mapping the input read sequences. Since MapCaller collects all information required for variant identification while reads are mapped onto the reference genome, variants can be called directly in the same process. Therefore, the conventional analysis pipeline can be simplified greatly. Most existing variant callers can only detect a few specific types of variants, however MapCaller can detect multiple types of variations, including SNVs, indels, inversions and translocations. We demonstrate that MapCaller not only produces comparable performance on variant calling, but it also spends much less time compared to some selected variant calling pipelines. MapCaller was developed under Linux 64-bit environment and implemented with standard C/C++. It takes read files (FASTA/FASTQ) as input and outputs all predicted variants in VCF format. The source codes of MapCaller and benchmark datasets are available at https://github.com/hsinnan75/MapCaller.

2 Methods

MapCaller aligns every short read onto a reference genome and collects the alignment information during read mapping to deduce sequence variants. MapCaller uses a modified algorithm of KART to perform read mapping. It also maintains a position frequency matrix to keep track of every

nucleobase's frequency at each position in the reference genome and collects all insertion and deletion events that are identified during read mapping. Furthermore, MapCaller learns all possible breakpoints from discordant or partial read alignments. Finally, MapCaller finds sequence variants based on all of the above-mentioned information. The novelty of our algorithm derives from the integration of read mapping and variation information gathering into a coherent system for variant calling.

2.1 Read mapping and alignment profiles

The details of read mapping method, KART can be found in our previous study. Here we focus on the high-level methodology description. KART adopts a divide-and-conquer strategy to handle matches and mismatches separately between read sequence and reference genome. KART identifies all locally maximal exact matches (LMEMs). We then cluster them according to their coordinates and fill gaps between LMEMs to create final alignments. In doing so, KART divides a read alignment into two groups: simple region pairs (abbreviated as *simple pairs*) and normal region pairs (*normal pairs*), where all simple pairs are LMEMs and normal pairs are gaps between simple pairs and might require gapped alignment (due to mismatches or indels). Thus, all SNVs and indels should be found in normal pairs, and breakpoints can be deduced from normal pairs at either end of read alignment. We will explain the identification metrics of each variant type below.

For the convenience of describing the methodology of MapCaller, we define the following notations. Given a read sequence R, the reference genome G, Let R_i be the i-th residue of R and $R[i_1, i_2]$ be the substring between R_{i1} and R_{i2}. Likewise, let G_j be the j-th nucleotide of G and $G[j_1, j_2]$ be the substring between G_{j1} and G_{j2}. A simple pair (or a normal pair) consists of a read's substring and its counterpart of reference's substring. They can be represented as $(R[i_1, i_2], G[j_1, j_2])$. One or more simple/normal pair forms a candidate alignment. We perform pairwise alignment for each normal pair in a candidate alignment.

We also check the alignment quality of a normal pair at either end of the read sequence to determine whether they should be removed. The quality evaluation is as follows. Given an alignment of a normal pair at either end, MapCaller counts the number of mismatches and state transitions of the alignment. A state transition is an alignment state change of base alignment, insertion and deletion. If there are more than three state transitions or the number of mismatches is more than $0.3N$ (N is the number of base alignments) in the normal pair, it will be removed from the candidate alignment. Such normal pair may appear due to false mapping or structural variants. MapCaller infers breakpoints based on such normal pairs.

Fig. 1 illustrates two cases of read alignments that reveal a translocation event and an inversion event respectively. In Fig. 1(A), paired-end reads Read1 and Read2 are mapped far away from each other due to a translocation event in the sample genome. Thus their alignments are considered discordant and we infer the corresponding breakpoints according to such cases. If a read alignment partially covers region A and partially covers region B, we will learn the occurrence of the breakpoint based on the discarded normal pairs. Likewise, in Fig. 1(B), paired-end reads Read3 and Read4 are mapped with same orientation due to an inversion event in the sample genome. Their alignments are also considered discordant and we infer the corresponding breakpoints according to such cases. We will learn the occurrences of breakpoints from read alignments with discarded normal pairs.

If two normal pairs at both ends are removed after quality evaluation, then we will discard the whole candidate alignment since it is very likely the candidate alignment is a false alignment. Finally, each candidate alignment is evaluated by their numbers of exact matches and mismatches. For

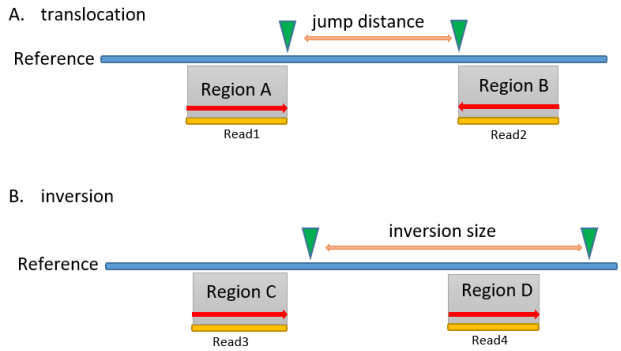

Figure 1. (A) Paired-end Read1 and Read2 are mapped distantly due to a translocation event in the sample genome. (B) Paired-end Read3 and Read4 are mapped with same orientation due to an inversion event in the sample genome.

each short read, we only keep the alignment with the highest alignment score.

MapCaller creates a position frequency matrix (PFM) to count the nucleobase frequencies at each position. PFM is a matrix of $4 \times L$, where L is the reference genome size. Therefore, each column of PFM represents the frequencies of nucleobases A, C, G and T at that position of the reference genome. MapCaller updates the nucleobase frequency at each column according to read alignments. Fig. 2 shows an example to illustrate how PFM works in this study. Five reads are mapped onto the reference genome. MapCaller counts the frequencies of each involved column. For example, PFM[3] = (1, 0, 0, 4) indicates there are one 'A' and four 'T's aligned at the third position of the reference genome. Any insertion and deletion events are kept otherwise. We use a 3-tuple, $Ins(Gpos, R[i,j], k)$ and $Del(Gpos, G[m,n], k)$ to represent an insertion and deletion event, where $Gpos$ indicates the occurrence location, $R[i,j]$ and $G[m,n]$ are the indel strings, and k is the number of occurrences. MapCaller would create the following three 3-tuples: Ins(3, T, 1), Del(5, C, 1) and Del(8, CG, 1) based on the example cases in Fig 2. We describe how the SNVs, indels, inversions and translocations are identified based on the PFM and by MapCaller below. It is noteworthy that the quality measurement of each type of variant is described in the Supplementary.

2.2 SNV detection

MapCaller uses PFM to keep track of the frequencies of the four nucleobases at each position in the reference genome. The depth of position p, denoted as $Depth(p)$, where $Depth(p)$ = PFM[p, A]+ PFM[p, C]+ PFM[p, G] + PFM[p, T]. We partition the reference genome sequence into blocks of 100 nucleotides. For each block i, MapCaller determines a threshold, denoted as $depthr(i)$, which is the half of the average depth for all the nucleotides within the block.

A nucleobase at position p is considered an alternative allele if its frequency is above $FreqThr(p) = Depth(p) \times MinAlleleFrequency$, where $MinAlleleFrequency$ is a user defined threshold (the default value is 0.2 for germline mutation and 0.02 for somatic mutation). Thus, a nucleobase b at position p is considered an SNV if the following two conditions are satisfied: (1) $Depth(p) \geq depthr(i)$; (2) PFM[p, b] $\geq FreqThr(p)$. Since the sample genome may carry multiple sets of chromosomes, the genotype at the same position can be homozygous or heterozygous. MapCaller considers both haploid and diploid scenarios. If only one nucleobase is called and its frequency is less than 1- $MinAlleleFrequency$, then the locus is considered heterozygous, otherwise it is homozygous.

2.3 Indel detection

read#1	A	T	T	T	G	C	G	T	C	G	A	…
read#2		T	T	-	C	C	G	T	-	-	A	…
read#3	A	T	T	-	G	-	G	T	C	G	T	…
read#4			T	-	G	C	G	A	C	G	A	…
read#5			A	-	G	C	G	A	C	G	A	…

PFM =

REF_{pos}	1	2	3	-	4	5	6	7	8	9	10	…
A	2	0	1	-	0	0	0	2	0	0	4	…
C	0	0	0	-	1	4	0	0	4	0	0	…
G	0	0	0	-	4	0	5	0	0	4	0	…
T	0	3	4	-	0	0	0	3	0	0	1	…

Figure 2. An example of position frequency matrix (PFM). Five read alignments are used to count the frequency of A, C, G and T at each position. Nucleobases in red are sequencing errors, and those in blue are SNPs.

MapCaller keeps track of all indel events using the 3-tuples, thus indel events can be deduced from those 3-tuples. We use *InsFreq*(*p*) and *DelFreq*(*p*) to represent the frequencies of insertion and deletion events at position *p*. However, the alignments involved with indels could be ambiguous. For example, the two following alignments produce identical alignment scores:

```
AGCATGCATTG        AGCATGCATTG
AGCAT----TG, and   AG----CATTG.
```

It can be observed that the two alignments will lead to different indel events, which are GCAT and CATG at neighboring positions. To avoid the ambiguity, MapCaller finds the indel with the maximal frequency within the range of (p -5, p + 5). MapCaller also counts the total frequencies of insertions and deletions (denoted as *InsFreq*(*p*) and *DelFreq*(*p*)) within the range as the representative frequency. MapCaller reports an insertion event at position *p* only if $InsFreq(p) \geq depthr(i) \times 0.25$. Likewise, it reports a deletion event at position *p* only if $DelFreq(p) \geq depthr(i) \times 0.35$. Since *depthr(i)* is normally lower than the neighboring depths when there are deletion events, we use 0.35 to determine the threshold for deletion detection.

2.4 Translocation and inversion detection

MapCaller estimates the average distance and fragment size of the paired-end reads periodically during read mapping. They are denoted as *AvgDist* and *AvgFragmentSize*. The definition of the distance and fragment size is illustrated in Fig. S1 in Supplementary. We use *AvgDist* to distinguish concordant pairs from discordant pairs. In Illumina sequencing protocol, two paired-end reads are supposed to be mapped to different orientations of the same chromosome within an expected distance. Concordant pairs match paired-end expectations, whereas discordant pairs do not. Fig. 3(A) shows some regular paired-end reads that are mapped concordantly since the sample genome do not have any structural variants at the corresponding region. Fig. 3(B) shows an example of a translocation event. The green dotted-rectangle is translocated from Breakpoint2 to Breakpoint1. Thus, the corresponding paired-end reads will be mapped discordantly. Likewise, Fig. 3(C) shows an example of an inversion event at BreakPoint3. The light-blue dotted-rectangle is the reversal of itself. Therefore, the corresponding paired-end reads will be mapped to the same chromosome strand.

MapCaller collects all discordant pairs and classifies them into translocation and inversion candidate groups. If paired-end reads are mapped to different orientation and their distance is greater than ($AvgDist \times 1.5$), they are put into translocation group, denoted as *TnlGroup*. If paired-end reads are mapped to the same orientation, they are put into inversion group, denoted as *InvGroup*. Since MapCaller collects all breakpoints from read alignments with a single-end clipping, we set a window to identify breakpoints with the most occurrences at each window. For each breakpoint *p*, we set a screening region of *AvgFragmentSize* width at both sides of *p*. To identify breakpoints of translocation events, we count the number of reads in *TnlGroup* that are mapped at the screening region. The number of reads at left side is denoted as *LRnum*(*p*), and the right side is denoted as *RRnum*(*p*). We also measure the average read depth at both sides, denoted as *Ldepth*(*p*) and *Rdepth*(*p*). If *p* is a true breakpoint for a translocation event, we should be able to observe a certain number of discordant read alignments in the *TnlGroup* that are separated by the breakpoint. Thus, we define *p* is a positive case if the following conditions are satisfied: (1) $LRnum(p) \geq depthr(i)$; (2) $LRnum(p) \geq Ldepth(p) \times 0.5$; (3) $RRnum(p) \geq depthr(i)$; and (4) $RRnum(p) \geq Rdepth(p) \times 0.5$. To identify breakpoints of inversion events, we adopt the same process to verify the positivity of every breakpoint by checking the discordant pairs in the *InvGroup*.

Table 1. The benchmark datasets. Sim_Ecoli, Sim_Chr1, and Sim_HG are synthetic datasets. SRR6062143 and SRR7781445 are generated from NA12878 (HG001); SRR3440404 and SRR6691661 are generated from NA24385 (HG002). TNL: translocation; INV: inversion.

Dataset	**# of short reads**	**read length**	**SNV**	**indels**	**TNL**	**INV**
Sim_Ecoli	1,472,676	100	12,822	1,035	6	3
Sim_Chr1	78,859,696	100	627,576	52,067	261	211
Sim_HG	978,158,892	100	8,003,103	666,213	2,650	2,559
SRR6062143	1,558,904,052	101	3,084,732	534,739	NA	NA
SRR7781445	609,362,230	151	3,084,732	534,739	NA	NA
SRR3440404	487,042,582	250	3,076,552	519,569	NA	NA
SRR6691661	638,722,972	151	3,076,552	519,569	NA	NA

3 Results

3.1 Experiment design

We develop a simulator to generate genome variations using the E.Coli K-12 strain, human chromosome 1 and human reference genome (GRCh38). The simulator (described in the Supplementary material *S8*) randomly generates sequence variations with occurrences of 2700 substitutions, 180 small indels (1~10 bp), 45 large indels (11~50 bp), 1 translocation (TNL, size ranges 1000~2000bp), 1 inversion (INV, size ranges 1000~2000bp) and 1 copy number variation (CNV, size ranges 300~1300bp) for every 1,000,000 base pairs. We use WGSIM (https://github.com/lh3/wgsim) to generate simulated short read sequences for each mutant genome. The simulated read coverage is 30X, sequencing error rate is 0.02, and read length is 100bp. The synthetic datasets are referred to as Sim_Ecoli, Sim_Chr1 and Sim_HG respectively. We also download four real NGS datasets from SRA web site, two from sample of HG001-NA12878 (RUN: SRR6062143 and SRR7781445) and two from sample of HG002-NA24385 (RUN: SRR3440404 and SRR6691661), where the dataset of SRR3440404 includes short reads from RUNs of SRR3440404 to SRR3440422 in order to have enough read depth. Note SRR6062143 and SRR7781445 are two separate runs of Illumina sequencing data of NA12878. SRR3440404 and SRR6691661 are also two separate runs of NA24385. GIAB (The Genome in a Bottle Consortium) provides high-confidence SNP, small indel calls for the two sample genomes. They can be found at ftp://ftp-trace.ncbi.nlm.nih.gov/giab/ftp/release/. Those variant calls were made by GATK, Freebayes and Sentieon. Table 1 shows the number of short reads of each dataset as well as the number of each type of variants. It is noteworthy that GIAB does not provide structural variant annotation for samples NA12878 and NA24385.

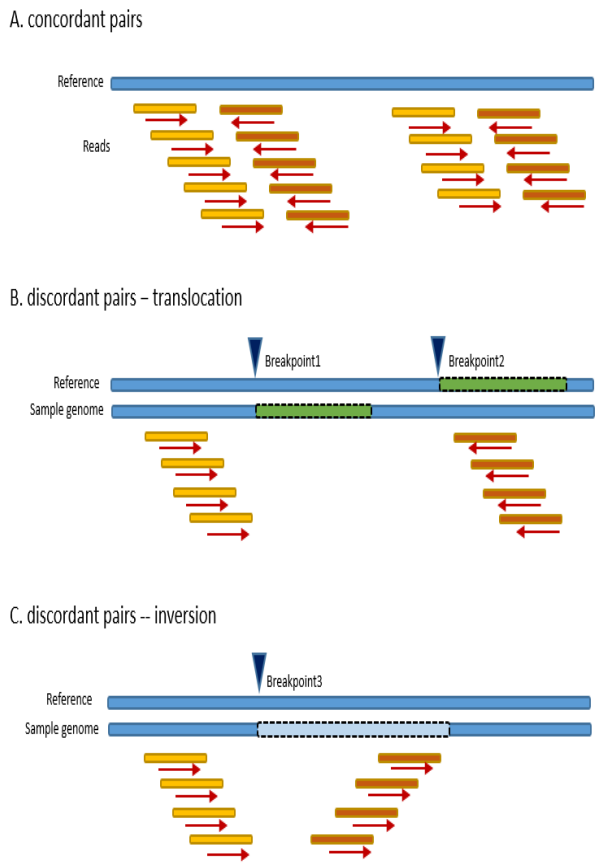

Figure 3. (A) Concordant pairs (B) discordant pairs −translocation (C) discordant pairs − inversion.

A conventional analysis pipeline includes a read mapper, SAM/BAM file processing, and a variant caller. Thus, in study, we compared the performance of MapCaller with different combinations of those read mappers and variant callers. For read mapping, we selected KART, BWA-MEM (BWA for short), Bowtie2, and GEM. For the variant calling, we selected Genome Analysis Tool Kit HaplotypeCaller (GATK for short), Freebayes and SAMtools mpileup (Mpileup for short). For SAM/BAM file processing, we used SAMtools view/sort to perform file format converting and alignment sorting. We also compared the performance of structural variant calling with existing methods, including DELLY(Rausch, et al., 2012), LUMPY(Layer, et al., 2014) , and SVDetect (Zeitouni, et al., 2010). The commands as well as the argument setting used for each pipeline are shown in Table *S*1 (Supplementary material).

It is noteworthy that since MapCaller integrates the algorithms of read mapping and variant calling, it handles the whole procedure of analysis pipeline alone. The run time was estimated from the read mapping to variant calling. We estimated the precision and recall for each dataset. We define a true positive case (TP) as a true variant call; a false positive case (FP) as a false variant call; and a false negative case (FN) as a true variant that is not called. A predicted SNV event is considered an TP if its genomic coordinate is correct without any tolerance, otherwise it is a FP. A predicted indel event is considered an TP if the genomic coordinate is within 10 bp, otherwise it is a FP.

Table 2. The performance comparison on synthetic datasets.

Pipeline	SNV		INDEL		Runtime (minutes)
	Precision	Recall	Precision	Recall	
Dataset: Sim_Ecoli					
MapCaller	100	99.3	99.9	98.9	0.1
BWA + Freebayes	99.6	99.5	100	98.7	9.6
BWA + Mpileup	100	99.2	100	98.7	3.5
BWA + GATK	99.3	76.4	100	15.9	66.4
Dataset: Sim_Chr1					
MapCaller	99.9	98.5	99.8	98.0	8.3
BWA + Freebayes	99.3	99.0	99.9	98.3	611.5
BWA + Mpileup	100	98.6	99.9	97.9	277.7
BWA + GATK	99.5	78.4	99.6	17.1	3264.2
Dataset: Sim_HG					
MapCaller	99.2	96.9	99.1	96.3	146.1
BWA + Freebayes	99.2	92.3	99.9	91.6	6954.7
BWA + Mpileup	44.7	91.8	99.9	91.0	3414.1
BWA + GATK	99.4	71.7	99.8	14.9	36812.3

3.2 Performance comparison on synthetic datasets

Table 1 summarizes the comparison result on the three synthetic datasets. We tested every combination of read mapper and variant caller. We found that BWA-MEM combined with any variant caller generally performed better than any other selected read mapper did; therefore, we only showed the performance of pipelines involved with BWA-MEM here to compare to MapCaller. The complete experiment result can be found in Supplementary (Table S2 and the excel file).

It can be observed that MapCaller and Freebayes performed comparably on the three synthetic datasets; however, MapCaller produced better performance on Sim_HG with respect to recalls. The SNV and indel recalls of MapCaller on Sim_HG are 96.9% and 96.3% respectively. When further analyzed some of the false negative cases, we found that most of false negatives occur at repetitive or highly similar regions. Freebayes and mpileup produced higher precisions on indel detection on Sim_HG; however, they compromised the recalls. The precisions and recalls of Freebayes are 99.9% and 91.6%, and those of mpileup are 99.9% and 91.0%. MapCaller produced balanced result on indel detection. Its precision and recall are 99.1% and 96.3% respectively.

Surprisingly GATK produced poor recalls on SNV and indel detection on all of the three datasets. The recalls on SNV detections were all below 80%, and those on indel detection were all below 20%. In this experiment, we fed all the three variant callers with the same read alignments; however, the fact that Freebayes and Mpileup produced much higher recalls on the three datasets suggested BWA could process those indel events correctly in the alignments. We tried to diagnose the reason by configuring the pipeline but it did not help. We then increased the read depth from 30X to 100X (Sim_Ecoli) and we found both of the recalls improved from 76.4% to 95.7% (SNV) and from 15.9% to 65.7% (indel). If we further increased the read depth to 200X (Sim_Ecoli), the two recalls achieve 96.7% and 87.2% respectively. It suggested GATK performs much better on NGS data with deeper coverage. The performance regarding different read depths is shown in Supplementary (Table S3). We also found that if the NGS data were error-free, then the performance of GATK would be very good. We simulated an NGS data without any sequencing errors (read depth: 30X, Sim_Ecoli) and tested with GATK combined with BWA, the precision and recall on SNV detection were 100% and 99.4% respectively,

and those on indel detection were 99.5% and 99.0% respectively. It implies that GATK is more sensitive to sequencing errors.

Another noteworthy observation is that Mpileup performed much worse on Sim_HG. The precision on SNV detection was only 44.7%. When we further analyzed the output, we found that Mpileup generated many false positive SNVs at loci with very shallow depth. Fortunately, we can filter out some of the false calls according to their confidence scores. We will investigate the relationship between variant calling performance and confidence score later.

In terms of run time, it can be observed that MapCaller is much more efficient than the selected pipelines. For example, MapCaller only spent 148 minutes to handle Sim_HG. However, the other three pipelines spent 3414, 6954, and 36812 minutes respectively on the dataset. Since the variant analysis pipeline consists of multiple steps, we can further decompose the runtime into multiple parts. Take the pipeline of BWA+GATK for example, it spent 1444 minutes for read mapping, 249 minutes for SAM/BAM file processing, and then another 35119 minutes for variant calling. If we only considered the run time for variant calling, MapCaller only spent 2.6 minutes (155 seconds). MapCaller is much more efficient because it not only adopts a very efficient read mapping algorithm, but it also saves run time by skipping processing SAM/BAM files and collects variant information directly during read mapping.

3.3 Performance comparison on real datasets

We downloaded four real NGS datasets sequenced for genomes of NA12878 (HG001) and NA24385 (HG002). However, the two genomes do not have ground truth variant annotation. GIAB provides reference calls that were made by GATK, Freebayes and Sentieon for the sample genomes. In this study, we used the reference calls as the gold standard to estimate the performance of each method.

Table 3 summarizes the comparison result. It can be observed that MapCaller, Freebayes and GATK performed comparably. It is not obvious that which method performed the best. Each method has their strength and weakness. For example, MapCaller and GATK generally produced higher precisions on SNV detections than the other two methods. Freebayes and GATK generally produced higher recalls on indel detections. However, it seemed like Mpileup produced relatively lower precisions on SNV detection and lower recalls on indel detection. Mpileup still generated many false positive cases on SNV detection. Its precisions were all below 20%. We also found that some of false calls made by MapCaller were due to ambiguous alignments of indels. If we relaxed the tolerance of locus difference for indel events to 50bp, the precisions and recalls of MapCaller on indel detection would be increased by around 10% and 4% respectively.

Moreover, some of false negative cases were due to poor alignment quality and their read alignments were discarded by MapCaller. For example, the reference calls reported three adjacent variants, which were chr1:20761541 (an insertion: AGAG), chr1:20761544 (SNV: C), and chr1:20761545 (a deletion: AT). Since there were more than three state transitions in the corresponding alignment, MapCaller chose not to report such variants. If two adjacent variants appear in two separated alignments, they could be still called by MapCaller. It is estimated NA12878 contains 11,018 indel events (2.1%) that are adjacent to one another within five nucleotides, and NA24385 contains 10,016 variants (2.0%). By contrast, MapCaller produced around 0.7% of indels that are adjacent within five nucleotides in average.

Since the reference calls were made by integrating three different callers, we analyzed the performance of overlapping calls by integrating MapCaller, Freebayes, Mpileup and GATK on the dataset of SRR6062143. If an SNV was called by all the four methods, the precision was 83.6%. If an indel was called by all the four methods, the precision was 79.2%. It suggested that the overlaps of variant calls could increase calling accuracy significantly. Moreover, we also found the union of the four callers could cover 99.9% of reference SNVs and 98.6% of reference indels. The performance with respective to overlaps among the four callers for each dataset was shown in Table S4 (Supplementary material).

Table 3. The performance comparison on real datasets.

Pipeline	SNV		INDEL		Runtime (hours)
	Precision	Recall	Precision	Recall	
Dataset: SRR6062143 (NA12878)					
MapCaller	76.2	98.8	66.4	89.2	2.0
BWA + Freebayes	70.8	96.2	61.3	91.8	122.7
BWA + Mpileup	22.3	97.4	70.1	73.5	70.2
BWA + GATK	77.1	97.3	67.1	95.5	203.8
Dataset: SRR7781445 (NA12878)					
MapCaller	78.6	97.3	66.2	88.3	1.5
BWA + Freebayes	72.1	96.1	58.3	95.0	122.3
BWA + Mpileup	17.8	97.4	68.0	71.4	71.8
BWA + GATK	77.3	97.2	63.9	96.0	161.8
Dataset: SRR3440404 (NA24385)					
MapCaller	76.4	98.3	66.0	89.8	2.0
BWA + Freebayes	67.3	95.8	60.1	95.5	116.4
BWA + Mpileup	8.5	99.7	66.8	73.1	60.9
BWA + GATK	77.1	99.7	68.7	95.7	227.9
Dataset: SRR6691661 (NA24385)					
MapCaller	77.5	97.9	60.7	91.3	1.5
BWA + Freebayes	70.4	95.8	58.1	94.8	96.5
BWA + Mpileup	16.8	99.4	66.6	76.2	49.7
BWA + GATK	78.2	99.8	63.2	99.0	157.1

In terms of run time, MapCaller was still much faster than any other pipeline. It is around 100 times faster than BWA+GATK. It spent around one or two hours to handle a human genome data with around 30X of read depth. If we only consider the run time for variant calling, MapCaller is much faster than GATK. For example, MapCaller spent three minutes on variant calling for SRR6062143, while GATK spent 203.8 hours for the same dataset. Though MapCaller runs very fast, it produces comparable result as Freebayes and GATK do in this analysis. Thus, we demonstrated MapCaller is a highly efficient variant calling method.

3.4 Variant filtering

We estimate the performance of MapCaller and each selected pipeline using the raw result. However, most variant callers provide specific filters to remove unlikely variants based on their algorithm design and statistic models; it may complicate the comparison if we try to optimize the filtering strategy for each caller. According to the performance comparison result using synthetic datasets, we found that MapCaller and the other selected methods produced high precisions on SNV and indel detection, except Mpileup lost its SNV accuracy on the dataset of Sim_HG. It suggested that MapCaller, Freebayes and GATK could produce reliable variant calls without any specific filters. However, Mpileup tends to produce more SNV calls. It is necessary to filter out false calls to improve the accuracy. Since all variant callers give an estimate on how likely a variant

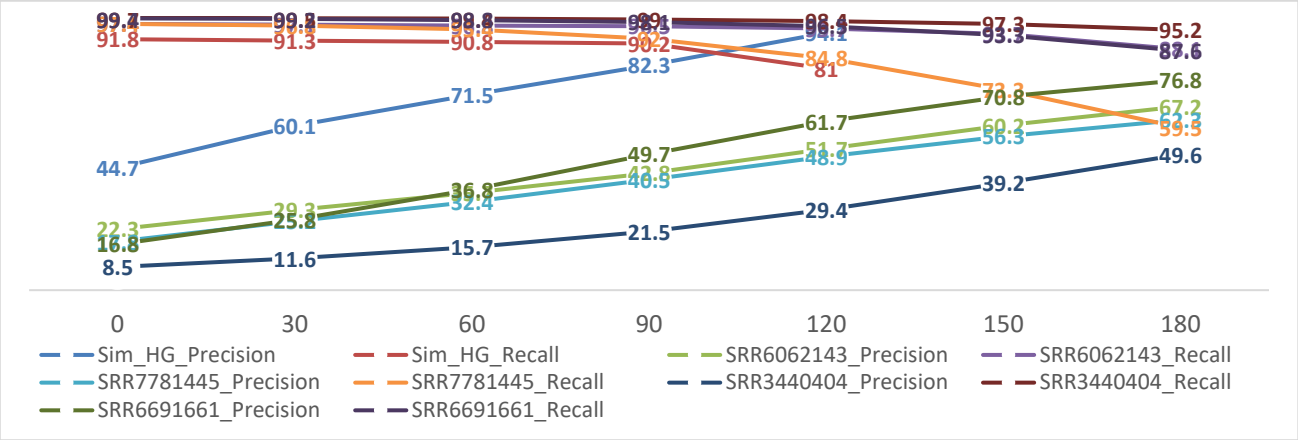

Figure 4. The performance evaluation for BWA+Mpileup with respective to different filtering threshold.

call is true with a quality score (the QUAL column), we analyzed the SNV accuracy regarding Mpileup on Sim_HG, SRR6062143, SRR7781445, SRR3440404 and SRR6691661 to investigate the relationship between precision/recall and QUAL values.

We filter out SNV calls with different QUAL thresholds. Fig 4 illustrates the analysis result. The QUAL threshold is from 0 to 180. Note that almost all of the quality scores of SNV calls for Sim_HG are below 150. It can be observed that the precision improves more and more when the QUAL threshold becomes higher and higher. On the other hand, the recall deteriorates at the same time. It is interesting that the quality scores of Mpileup are not consistent with the accuracy. For example, if the QUAL threshold is 90, we could observe the SNV accuracy on the five datasets varies a lot. It is much higher on Sim_HG, but it is lower on the real datasets. The highest precision is 82.3 while the lowest is only 21.5%. It implies the quality scores do not actually reflect the calling confidence.

Table 4. *Performance comparison on structural variant detection*

Data set	Method	Translocation		Inversion	
		Precision	Recall	Precision	Recall
Sim_Ecoli	MapCaller	100	100	100	100
	LUMPY	100	66.7	100	100
	SVDetect	100	100	100	100
	DELLY	NA	NA	100	100
Sim_Chr1	MapCaller	100	94.3	100	87.7
	LUMPY	84.2	33.3	99.5	92.9
	SVDetect	27.2	97.7	64.8	93.8
	DELLY	NA	NA	96.9	95.7
Sim_HG	MapCaller	99.9	94.8	100	87.8
	LUMPY	87.9	35.3	100	93.7
	SVDetect	28.9	95.4	62.3	93.1
	DELLY	NA	NA	99.1	95.2

3.5 Performance comparison on structural variant detection

In addition to SNVs and indels, MapCaller is also capable of identifying structural variants (referred to as SVs in the following description) simultaneously. We do not identify CNVs in this study. CNV calling will be included in MapCaller in the future version. Table 4 shows the performance comparison on SV detection for MapCaller and other selected callers. We estimated the precision and recall of each caller. It can be observed that MapCaller produced high precisions and recalls on all the three simulated datasets. In particularly, MapCaller produced the highest precisions among these callers. LUMPY produced high precisions and recalls on inversion detections. However, it produced relatively lower precisions and recalls on translocation detections. SVDetect lost precisions on translocation and inversion detection, but it generated much higher recalls on the two types of structural variants. Similar to LUMPY, DELLY generally produced high precisions and recalls on inversion detections, but it could not detect any translocation events on the three datasets. In summary, MapCaller was the only method that handled both translocation and inversion detection at high precision and recall.

4 Conclusions

In this manuscript, we present MapCaller, an integrated system for read mapping and variant calling. MapCaller collects read alignment information during read mapping and maintains a position frequency matrix to keep track of alignments at each position of the reference genome. We evaluate the performance of MapCaller and the selected variant calling pipelines using three synthetic datasets and four real data sets from human genomes. The comparison results show that MapCaller not only identifies highly accurate variants, but it also spends the least amount of time. MapCaller is also versatile. It is capable of identifying SNVs, INDELs and structural variants simultaneously.

All current variant calling pipelines require SAM/BAM files as inputs and take multiple steps to rearrange the alignments for variant calling. It is not only time consuming, but also requires a huge amount of disk space. MapCaller identifies variants directly from the NGS short reads. Thus, it is not necessary to output SAM/BAM files and it saves a lot of disk space. Pipelines involved with GATK require even more disk space for the additional preprocess of alignments. Thus, MapCaller is both time and space efficient. In particularly, MapCaller is very user-friendly. It takes only one command line to handle read mapping and variant calling. Since more and more medical and biological researches adopt NGS technologies to characterize the genetic variations between individuals, we believe MapCaller is a fast and reliable variant calling method.

Acknowledgements

Funding: This work was supported by Bioinformatics Core Facility for Translational Medicine and Biotechnology Development/Ministry of Science and Technology (Taiwan) 105-2319-B-400-002.

Conflict of Interest: none declared.

References

Carver, E.A. and Stubbs, L. Zooming in on the human-mouse comparative map: genome conservation re-examined on a high-resolution scale. *Genome Res* 1997;7(12):1123-1137.

DePristo, M.A., *et al.* A framework for variation discovery and genotyping using next-generation DNA sequencing data. *Nat Genet* 2011;43(5):491-+.

Dong, L., *et al.* Clinical Next Generation Sequencing for Precision Medicine in Cancer. *Curr Genomics* 2015;16(4):253-263.

Erik Garrison, G.M. Haplotype-based variant detection from short-read sequencing. *arXiv preprint* 2012(arXiv:1207.3907):q-bio.GN.

Hwang, S., *et al.* Systematic comparison of variant calling pipelines using gold standard personal exome variants. *Sci Rep-Uk* 2015;5.

Kim, D., Landmead, B. and Salzberg, S.L. HISAT: a fast spliced aligner with low memory requirements. *Nature methods* 2015;12(4):357-U121.

Koboldt, D.C., *et al.* VarScan: variant detection in massively parallel sequencing of individual and pooled samples. *Bioinformatics* 2009;25(17):2283-2285.

Ku, C.S. and Roukos, D.H. From next-generation sequencing to nanopore sequencing technology: paving the way to personalized genomic medicine. *Expert Rev Med Devic* 2013;10(1):1-6.

Kulkarni, P. and Frommolt, P. Challenges in the Setup of Large-scale Next-Generation Sequencing Analysis Workflows. *Comput Struct Biotec* 2017;15:471-477.

Langmead, B. and Salzberg, S.L. Fast gapped-read alignment with Bowtie 2. *Nature methods* 2012;9(4):357-359.

Langmead, B., *et al.* Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome biology* 2009;10(3):R25.

Layer, R.M., *et al.* LUMPY: a probabilistic framework for structural variant discovery. *Genome Biol* 2014;15(6).

Li, H. and Durbin, R. Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics* 2009;25(14):1754-1760.

Li, H., *et al.* The Sequence Alignment/Map format and SAMtools. *Bioinformatics* 2009;25(16):2078-2079.

Liao, Y., Smyth, G.K. and Shi, W. The Subread aligner: fast, accurate and scalable read mapping by seed-and-vote. *Nucleic Acids Res* 2013;41(10).

Lin, H.N. and Hsu, W.L. Kart: a divide-and-conquer algorithm for NGS read alignment. *Bioinformatics* 2017;33(15):2281-2287.

Marco-Sola, S., *et al.* The GEM mapper: fast, accurate and versatile alignment by filtration. *Nat Methods* 2012;9(12):1185-U1176.

Rausch, T., *et al.* DELLY: structural variant discovery by integrated paired-end and split-read analysis. *Bioinformatics* 2012;28(18):I333-I339.

Rimmer, A., *et al.* Integrating mapping-, assembly- and haplotype-based approaches for calling variants in clinical sequencing applications. *Nat Genet* 2014;46(8):912-918.

Yen, J.L., *et al.* A variant by any name: quantifying annotation discordance across tools and clinical databases. *Genome Med* 2017;9.

Zeitouni, B., *et al.* SVDetect: a tool to identify genomic structural variations from paired-end and mate-pair sequencing data. *Bioinformatics* 2010;26(15):1895-1896.