Published in final edited form as:
J Parallel Distrib Comput. 2012 January ; 72(1): 83–93. doi:10.1016/j.jpdc.2011.08.001.

High Performance Multiple Sequence Alignment System for Pyrosequencing Reads from Multiple Reference Genomes

Fahad Saeed[*], **Alan Perez-Rathke**[†], **Jaroslaw Gwarnicki**[†], **Tanya Berger-Wolf**[†], and **Ashfaq Khokhar**[‡]

[†]Department of Computer Science, University of Illinois at Chicago, IL USA

[‡]Department of Electrical and Computer Engineering, University of Illinois at Chicago, IL USA

Abstract

Genome resequencing with short reads generated from pyrosequencing generally relies on mapping the short reads against a single reference genome. However, mapping of reads from multiple reference genomes is not possible using a pairwise mapping algorithm. In order to align the reads w.r.t each other and the reference genomes, existing multiple sequence alignment(MSA) methods cannot be used because they do not take into account the position of these short reads with respect to the genome, and are highly inefficient for large number of sequences. In this paper, we develop a highly scalable parallel algorithm based on domain decomposition, referred to as P-Pyro-Align, to align such large number of reads from single or multiple reference genomes. The proposed alignment algorithm accurately aligns the erroneous reads, and has been implemented on a cluster of workstations using MPI library. Experimental results for different problem sizes are analyzed in terms of execution time, quality of the alignments, and the ability of the algorithm to handle reads from multiple haplotypes. We report high quality multiple alignment of up to 0.5 million reads. The algorithm is shown to be highly scalable and exhibits super-linear speedups with increasing number of processors.

1 Introduction

For over a decade, Sanger sequencing has been the cornerstone of genome sequencing including that of microbial genomes. Improvements in DNA sequencing techniques and advances in data storage and analysis, as well as developments in bioinformatics have reduced the cost to a mere $8000 – $10000 per megabase of high quality genome draft sequence. However, the need of more efficient and cost effective approaches has led to the development of new sequencing technologies such as the 454 GS20 sequencing platform. It is a non-cloning pyrosequencing based platform that is capable of generating 1 million overlapping reads in a single run. However, multitude of factors, such as relatively short read lengths as compared to Sanger (as of year 2010: an average of 100–400 nt length reads compared to 800–1000 nt length reads for Sanger sequencing), lack of a paired end protocol, and limited accuracy of individual reads for repetitive DNA, particularly in the case of monopolymer repeats, present many computational challenges [1] to make pyrosequencing useful for biology and bioinformatics applications.

One of the most widely employed pre processing step for many applications, including haplotype reconstruction [2] [3], analysis of microbial community analysis [4], analysis of genes for diseases [5], is the alignment of these reads with the wild type. For important

[*]The author is currently affiliated with the National Heart Lung and Blood Institute (NHLBI), National Institutes of Health (NIH) Bethesda MD, USA. This work was completed when Fahad Saeed was a PhD candidate at the Department of Electrical and Computer Engineering, University of Illinois at Chicago, IL USA

applications such as viral population estimation or haplotype reconstruction of various viruses e.g., HIV in a population, scientists usually have the information about the wild type genome of the virus. To this date, numerous tools have been suggested for mapping short reads to the single reference genome. These methods include strategies for indexing substrings of short reads or references with the use of k-mer counts or spaces seeds. The list of academic tools available for such mapping inlcude Bowtie, BWA, CloudBurst, MAQ, MOM, MosaikAligner, mrFAST, mrsFAST, Pash, PASS, PatMaN, RazorS, RMAP, SeqMap, SHRiMP, SliderII, SOAP, SOAP2, ssaha2 [6–23], and commercial tools such as ZOOM [24]. The current demand in alignment can be met with new indexing strategies and efficient data processing [25]. However, these strategies are usually at the cost of simplifying the mapping problem and not allowing complex alignments, including gaps or alignment with multiple reference genomes.

Natural strains of genomes that have a high level of individual difference, such as natural inbred strains of Arabidopsis or divergent strains of HIV, pose a considerable alignment challenge [26]. It has been shown in [27, 28] that several percent of reference genome can be missing or very divergent in distinct strains of different species. This poses a problem for pairwise mapping algorithms because the mappings can result in regions that are inaccessible when aligned to the wildtype of the species. This is particularly true for algorithms that do not take into account mismatches and gaps. Fig 1 shows an example set of reads from different strains of genomes from the same species.

Apart from the above mentioned difficulties in using pairwise mapping algorithms for short reads, aligning against only a single reference has its own shortcomings. Aligning against a single reference biases the analysis towards a comparison within the sequence space highly conserved with the reference. Taking into account all the strains of the genome reduces this bias. Aligning each read individually with each reference genome loses important information such as variation SNP's in different strains, besides increasing computation time and memory. Therefore, to interpret the alignment results some kind of merging and interpreting strategy is required.

One such strategy was recently introduced by Schneegerber et al. in [26], called GenomeMapper. It performs simultaneous alignment of short reads against multiple genomes using a hash-based data structure [26, 29]. However, the accuracy of such alignments would be limited by the graph structure that captures the information from different strains of the same species. Also the method has the drawback that the reads are *pairwise* aligned to the graph structure and the comparison of reads with each other is not achieved. In this paper, we present a solution to the problem of aligning pyroreads from multiple genomes using a *multiple alignment* methodology on multiprocessor platforms.

In theory, alignment of multiple sequences can be achieved using pair-wise alignment, each pair getting alignment score and then maximizing the sum of all the pair-wise alignment scores. But for optimal alignment the sum of all the pair-wise alignment scores needs to be maximized, which is an NP complete problem [30]. Towards this end, dynamic programming based solutions of $O(L^N)$ complexity have been pursued, where N is the number of sequences and L is the average length of a sequence. Such accurate optimizations are not practical for large number of sequences -as is the case in pyrosequencing, thus making heuristic algorithms as the only feasible option. The literature on these heuristics is vast and includes widely used works, such as Notredame et. al. [31], Edgar [32], Thompson et. al. [33], Do et. al. [34], and Morgenstern et.al. [35].

Recently we introduced a multiple alignment system for pyrosequencing reads, known as pyroalign [36]. Although, the sequential algorithm is highly efficient compared to the other

alignment algorithms, it is still not feasible to multiple align large number of reads from multiple genomes on a single processor. It takes over 7 days to *multiple align* 200,000 pyroreads on single processor using this best known sequential algorithm [36], and the time increases drastically with the increase in number of reads.

In this paper, we present a parallel *multiple alignment* solution for aligning reads from *multiple genomes*. The objective of our work is to develop a high performance multiple alignment system for small error prone reads from multiple genomes strands of the same species, such that the errors in the alignment are 'highlighted' and the system is able to handle large number of reads, as may be expected from pyrosequencing procedure. We emphasize that this problem is considerably harder and different compared to mapping of the reads to a single reference genome.

The multiple sequence alignment algorithm presented, distinct from pairwise mapping or genome reconstruction, is specifically designed for the alignment of reads from multiple genomes from different strains of the same species. The proposed algorithm is a combination of the domain decomposition concept introduced in [37] [38] and the exploitation of pyroreads characteristics for parallelization, therefore it is capable of aligning very large number of pyroreads. It takes into account the position of the reads with respect to the reference genomes, and assigns weight to the leading and trailing gaps for the reads. After the decomposition, the reads are distributed among multiple processors using a metric calculated from the reads. The reads are multiple aligned on each processor with respect to each other and the genome. Then, the aligned reads on the processors are merged by exploiting the characteristics specific to pyro-reads. We assume that the reads may be generated from one or many genomes, with 'forward' orientation. We also assume that a wild-type of the species is available, as is generally the case for resequencing experiments. In our experiments, we have used HIV-pol gene virus as the reference genome (with length of 1971bp) and simulator Readsim [39] to generate these reads. Since the complexity for multiple alignments is dominated by the number of reads, the length of the genome has minimal effect in terms of memory and timings. Please note that the above genome is only used as a proof of concept for the technique. The importance of aligning reads from multiple genomes is discussed in the literature [29] [26] and is not stated here in the interest of brevity. We are able to align 200, 000 reads on a 24 processor system in just 278 minutes. The alignment of same number of reads takes approximately 7 days on a single node system.

For the sake of completeness, let's first formally define the multiple sequence alignment problem in its generic form, without indulging with the issues such as scoring functions. Let N sequences be presented as a set $S = \{S_1, S_2, S_3, \cdots, S_N\}$ and let $S' = \{S'_1, S'_2, S'_3, \cdots, S'_N\}$ be the aligned sequence set, such that all the sequences in S' are of equal length, have maximum overlap, and the score of the global map is maximum according to some scoring mechanism suitable for the application. A perfect multiple alignment for pyroreads would be, that the reads are aligned with each other such that the position of the reads with respect to the reference genome(s) is conserved; the reads have maximum overlap (for conserved genome regions) and are of equal lengths after the alignment, including leading and trailing gaps.

2 Proposed Parallel Multiple Sequence Alignment Algorithm for Pyroreads: P-Pyro-Align

The intuitive idea behind the proposed *P-Pyro-Align* algorithm is to first place the reads in correct orientation with respect to the wildtype and then use progressive alignment to achieve the final alignment. The starting position of the reads w.r.t the genome is referred to

as *s-rank* and is used to redistribute the reads on multiple processors. Regular sampling [40] based on *s*-rank is used to achieve load-balancing among multiple processors. The final alignment is produced by combining the alignment produced on each processor. To produce the global alignment we exploit the characteristics of the pyroreads and bring in techniques from data structures and parallel computing such as histogram mapping of indel positions on each processor, sampling based load-balancing and many-to-many communications to realize a low complexity, highly scalable solution in terms of time, memory and communication.

First in Fig. 2 we present an outline of the proposed parallel alignment algorithm, *P-Pyro-Align*. This outline is followed by a detailed description of the salient steps of the algorithm.

2.1 Semi-Global Alignment

The first step is to determine the position of each read with respect to the reference genome. If this step is omitted, there are number of alignments that would be correct, but would be inaccurate if analyzed in the global context. A read that is not constricted in terms of genomic position, may give the same score (SP score) for the multiple alignment but would be incorrect in context of the reference. To accomplish the task of 'placing' the reads in the correct context with respect to the reference genome we employ semi-global alignment procedure [1].

The semi global alignment is also referred to as overlapping alignment because the sequences are globally aligned ignoring the start and end gaps. For semi-global alignment we use a well known modified version of Needleman-Wunsch algorithm [41] described in [42]. The modification in the basic version of Needleman-Wunsch is required to ignore the leading and trailing gaps of the reads when aligning to the reference genome. If the leading and trailing gaps are not ignored, considering the short length of the reads, the alignment scores would be dominated by these gaps, hence giving an inaccurate alignment with respect to the genome.

Let the two sequences to be aligned be s and t of length m and n; and $M(i, j)$ presents the score of the optimal alignment. Since, we do not wish to penalize the starting gaps, we modify the dynamic programming matrix by initializing the first row and first column to be zero. The gaps at the end are also not to be penalized. Let $M(i, j)$ represent the optimal score of $s_1, \cdots, s_i$ and $t_1, \cdots, t_j$. Then $M(m, j)$ is the score that represents optimally aligned s with $t_{1, \cdots, j}$. The optimal alignment therefore, is now detected as the maximum value on the last row or column. Therefore the best score is $M(i, j) = max_{k,l}(M(k, n), M(m, l))$, and the alignment can be obtained by tracking the path from $M(i, j)$ to $M(0, 0)$. For additional details on semi-global alignment we refer the readers to [42].

Once each read has been semi-globally aligned with the wildtype genome on each processor, we obtain reads with leading and trailing gaps, where the first character after the gaps is the starting position of the read with respect to the wildtype genome.

2.2 Domain Decomposition of the Reads

After the reads are aligned with respect to the wildtype, on each processor we have the information of the reads position. This read position, referred to as *s-rank*, is important for realizing correct alignment. We also exploit this information to decompose our domain of

[1]This can be replaced with an indexing methodology for large genomes; for multiprocessors, in favor of simplicity, we employ simple semi-global alignment which can be enhanced using indexing

pyro-reads on multi processor platforms. The decomposition is similar to our generic strategy for multiple alignment [37].

Since we are assuming N/p random sequences on each processor; each processor may have reads with starting positions from the start to the end of genome. However, we aim to reallocate the reads for progressive alignment such that the reads that are closer to each other in terms of *s-rank* are clustered on a single processor.

Progressive alignments are produced in each processor by combining pairwise alignments of most similar sequences. All progressive alignment methods require two stages: first stage in which the relationships between the sequences are represented as a tree, called a guide tree, and second stage in which multiple sequence alignment(MSA) is built by adding the sequences sequentially to the growing MSA according to the guide tree.

The method followed by most of the progressive multiple alignment algorithms is that first a pairwise similarity measure is computed for each pair of sequences and this measure is organized in a matrix form. A tree is constructed from this distance matrix using Unweighted Pair Group Method with Arithmetic Mean (UPGMA) or neighboring joining. The progressive alignment is thus built, following the branching order of the tree, giving a multiple alignment. These steps require $O(N^2)$ time each, where N is the number of reads.

In the case of pyrosequencing, we exploit the fact that the reads are coming from *nearly* the same reference in terms of species. This in turn implies that the reads starting from the same or near same 'starting' point with respect to the reference genome are likely to be similar to each other. Therefore, we already have the ordering information or the 'guide tree'.

Let there be N/p number of reads in each processor $R = R_1, R_2, \cdots, R_{N/p}$ generated from the reference genome of length L_G. Also, let the length of each read be denoted by $L(R_p)$. After executing semi-global alignment using the algorithm discussed in the previous section, let each read be presented by R_{pq}, where the pth read has q leading gaps and $L_G - q - L(R_p)$ trailing gaps. Then the reordering algorithm would reorder the reads such that the read R_{pq} comes 'before' $R_{(p+1)(q+x)}$, $1 \leq q \leq L_G$ and $1 \leq x \leq L(R_p)$.

In an ideal load-balancing situation the number of reads on each processor would be the range of the s-ranks divided by the number of processors. However, the coverage of each position in pyro-sequencing is not constant and can vary considerably which would lead to unbalanced load on processors. Therefore, we employ regular sampling [40] based distribution of reads on multiple processors. For the sake of completeness, the analysis of the distribution of the reads using regular sampling is extensively discussed in section 3.

2.3 Pair-wise and Profile-Profile Alignments

The ordering and the distribution of the reads on multiple processors determined in the preceding step is now used to conduct the progressive alignment. The procedure described here is executed on each of the processors independently after the reads have been distributed according to the s-ranks. Traditional progressive alignment requires that the sequences most similar to each other are aligned first. Thereafter, sequences are added one by one to the multiple alignments determined according to some similarity metric. In order to devise a low complexity system, we presented a hierarchical progressive alignment procedure in [36]. However, our experiments with extremely large data sets for alignment suggest that this hierarchical strategy does not work accurately due to weighted leading and trailing gaps. The parallel progressive alignment procedure is formulated as follows.

On each processor, pair-wise local alignment using standard Smith-Waterman is executed on each overlapping pair of reads (the ordering is still the same as discussed in the previous section). After this stage, the reads are aligned in pairs such that we have $\frac{N}{2p}$ pairs of aligned reads on each processor. These $\frac{N}{2p}$ pairs of reads are then used for profile alignments as discussed below.

Profile-profile alignments are used to re-align two or more existing alignments (in our case the pairs of aligned reads). It is useful for two reasons; one being that the user may want to add sequences gradually, and second being that the user may want to keep one high quality profile fixed and keep on adding sequences aligned to that fixed profile [33].

In this stage of the algorithm, the $N/2p$ pairs of aligned reads have to be combined to get a multiple alignment. The paired profiles are then combined one by one, with reads added in ascending order of s-ranks.

In order to apply pair-wise alignment functions to profiles, a scoring function must be defined, similar to the substitution methods defined for pair-wise alignments. Profile sum of pairs (PSP) is the function used in Clustalw [33], Mafft [43] and Muscle [44] to maximize Sum of Pairs (SP) score, which in turn maximizes the alignment score such that the columns in the profiles are preserved. Profile functions have evolved to be quite complex and good discussion on these can be found at [44–46] and [47]. For our purposes we use the profile functions from the clustalw system.

2.4 The Merging Strategy

The ordering and alignment of the reads with weighted leading and trailing gaps on each processor gives a very high quality alignment. The important question is how to merge independently aligned reads on different processors into a single alignment in an efficient manner such that the computations required are minimal and communication between processors is far less than the computational load.

The merging for general multiple alignment on multiple processors has been introduced for domain decomposition strategy in [37]. We further modify the technique introduced in [37] to give a low complexity merging solution for pyrosequencing reads in terms of computational and communication load.

We observe that after the alignment performed in each processor, large columns of gaps are introduced over the entire range because of the insertions and deletions in pyrosequencing reads. This is precisely the only information required to merge two high quality alignments. Since the reads are coming from the same reference species, a large column of indels in one alignment and absence of any in the other alignment implies that the indels in the first alignment are due to insertions and/or deletions in those reads. Thus, insertion of large column of indels in the second alignment would make a perfect global alignment for pyro-reads. The concept is illustrated in Fig. 3. The important point to note here is, that unlike traditional profile-profile alignment, the only information that is shared among processors is the position of the large indels in the alignment on individual processors. This small amount of data is communicated among the processors involved in the merging. An example final alignment from the *P-Pyro-Align* algorithm can be seen in Fig. 4.

3 Analysis of Computation and Communication Costs

For the computation and communication analysis we use a coarse grained computing model such as C^3-model [48] and [49]. Also, for analysis purposes, we assume that the Clustalw System [33] is being used at each processor as the underlying profile-profile alignment

system. It must be noted that the computation complexity of the alignment step will vary depending on the sequential MSA system used for alignment within each processor.

In the following analysis we assume that each processor has $w = N/p$ pyro-reads, where N is the total number of reads to be aligned, and p is the number of processors. The average length of a read is L_R and the length of the genome is L_G. In Table 1, we outline the computation cost of each step of the algorithm.

3.1 Communication Cost

The communication overhead is an important factor that dictates the performance of a distributed message passing parallel system. If the communication overhead is much higher than the computation cost, the performance of the system is limited. Fortunately, the communication cost of our system is much less than the cost of the alignment. Essentially, the proposed P-Pyro-Align algorithm has three rounds of communication. In the first round, a small set of samples is collected at the root processor and a set of pivots based on s-ranks is broadcast from the root processor. In the second round, reads are redistributed to achieve better alignments and balanced load distribution. In the third round, the location of large indel columns from each processor are broadcast to all the other processors. For the analysis of the communication costs we have adopted the coarse grained computation model [48] [50] and domain decomposition communication cost analysis [37]. However, we ignore the message start up costs and assume unit time to transmit each data byte.

We have assumed the Regular Sampling strategy [40] because of its suitability to our problem domain. Some of the reasons are:

1. The strategy is independent of the distribution of original data, compared to some other strategies such as Huang and Chow [51].
2. It helps in partitioning of data into ordered subsets of approx. equal size. This presents an efficient strategy for load balancing as unequal number of sequences on different processors would mean unequal computation load, leading to poor performance. In the presence of data skew, regular sampling guarantees that no processor computes more than ($2\frac{N}{p}$) sequences [40].

   It has been shown in [40] that regular sampling yields optimal partitioning results as long as $N > p^3$, i.e., the number of data items N is much larger than the number of processors p, which would be a normal case in the MSA application.

   The partitioning of data can also be achieved by dividing the length of the genome with the number of processors and distributing the reads according to the pivot from the division. However, distributing randomly positioned reads w.r.t the genome from pyrosequencing can create unbalanced computational loads on multiprocessors. Hence partitioning the data trivially does not give any theoretical guarantees for load distribution making the process more random and unreliable.

3.1.1 First Communication Round—Assuming $k = p - 1$, i.e., each processor chooses $p - 1$ samples, the complexity of the first phase is $O(p^2 L_R) + O(p \log p) + O(k \times p \log p)$, where $O(p^2 L_R)$ is the time to collect $p(p-1)$ samples of average length L_R at the root processor, $O(p \log p)$ is the time required to broadcast $p-1$ pivots to all the processor and ($k \times p \log p$) is the time required to broadcast $k \times p$ sequences to all the processors.

3.1.2 Second Communication Round—In the second round each processor sends the sequences having s-rank in the range of bucket i to processor i. Each processor partitions its block of reads into p sub-blocks using pivots as bucket boundaries, and sends sub-block i to

processor i.. The sizes of these sub-blocks can vary from 0 to $\frac{N}{p}$ sequences depending on the initial data distribution. Taking the average case where the elements in the processor are distributed uniformly, each sub-block will have $\frac{N}{p^2}$ sequences. Thus this step would require $O(\frac{N}{p})$ time assuming an all-to-all personalized broadcast communication primitive [49]. However, in the following we show that based on regular sampling no processor will receive more than $2\frac{N}{p}$ elements in total in the worst case. Therefore still the overall communication cost will be $O(\frac{N}{pL_R})$.

Let's denote the pivots chosen in the first phase by the array: $y_1, y_2, y_3, \cdots, y_{p-1}$. Consider any processors i, where $1 < i < p$. All the sequences to be processed by processor i must have s-rank $> y_{i-1}$ and $\leq y_i$. There are $(i-2)p+\frac{p}{2}$ sequences of the regular sample which are $\leq y_{i-1}$, implying that there are at least $lb=((i-2)p+\frac{p}{2})\frac{N}{p^2}$ sequences in the entire data that have s-rank $\leq y_{i-1}$. On the other hand, there are $(p-i)p-\frac{p}{2}$ sequences in the regular sample that have s-rank $> y_i$. Thus, there are $ub=((p-i)p-\frac{p}{2})\frac{N}{p^2}$ sequences of N which are $> y_i$. Since the total number of sequences is N, at most $N-ub-lb$ sequences will get assigned to processor i. It is easy to show that this expression is upper bounded by $2\frac{N}{p}$. The cases for $i = 1$ and $i = p$ are special because the pivot interval for these two processors is $\frac{p}{2}$. The load for these processors will always be less than $2\frac{N}{p}$. Due to page limitations, we refer to [40] for further details of the analysis.

3.1.3 Third Communication Round—In the third round each processor sends the position of long column indels in the alignments. In the analysis we assume a worst case scenario where a long column of indels is encountered at each alternating position i.e. after each column of DNA nucleotides a column of indels is encountered in the alignment. We will assume that the columns from each processors are broadcasted to every other processor. The length of the columns are equal to the number of reads N because of obvious reasons and the number of columns would be equal to L_G as an upper bound. Then the communication cost would be equal to $O(NL_G \log p)$.

The total asymptotic time complexity T of the algorithm is shown in equation 3.

4 Performance Analysis

The performance evaluation process has been divided into two parts: the first part deals with the quality assessment, and the second part deals with traditional HPC metrics such as execution time, scalability, memory requirements, etc. The performance evaluation of the P-Pyro-Align algorithm is carried out on a Beowulf Cluster consisting of 24 Intel Xeon processors, each running at 3.0GHz, with 512KB cache and 2GB DRAM memory. As for the interconnection network, the system uses Intel Gigabit network interface cards on each cluster node. The operating system on each node is RedHat Linux 4.2 (Kernel level: 2.6.9-22.ELsmp).

4.1 Experimental Setup and Quality Assessment

As discussed earlier in the paper, the exact solution for multiple alignment is not feasible and heuristics are employed. Most of these heuristics perform well in practice but there is generally no theoretical justification possible for these heuristics [52]. For P-Pyro-Align it can be shown that the semi-global alignment of the reads with the reference genome is analogous to center star alignment. The center star alignment is shown to give results within 2-approx of the optimal alignment [52] in worst case and same can be expected from the

semi-global alignment of reads with reference genome. The accuracy of the later stages is confirmed by rigorous quality assessment procedure described in the section below.

The assessment of the quality is done in the spirit of determining how good the multiple alignment of the reads is. Aligning reads from multiple genomes have been shown to increase the number of recovered SNP's by 15%, of deletions by 22.6% and insertions by 37.2% [26]. However, our quality assessment is to determine how good the *multiple alignment* is. Clearly, the advantages discussed for using multiple genomes over single reference would be evident if the alignment quality obtained is good [26].

To investigate the quality of the alignment produced by the algorithm we have used Readsim simulator [39] to generate the reads. The quality assessment of multiple alignment is generally carried out using benchmarks such as Prefab [32] or BaliBase [53]. However, these benchmarks are not designed to assess the quality of the aligned reads produced from pyrosequencing, and there are no benchmarks available specifically for these reads. Therefore, a system has to be developed to assess the quality of the aligned reads. The experimental setup for the quality assessment of the alignment procedure is shown in Fig. 5 and is explained below.

Our quality assessment has two objectives: (1) to assess the quality of the alignment produced by P-Pyro-Align with respect to the original genome, and (2) to ensure that the system is able to handle reads from multiple genomes for alignment.

We used a HIV pol gene virus with length of 1971bp as the wildtype for the experiments. The wildtype is then used to produce 6 sets of genomes, randomly mutated at different rates. The six sets of genomes are Dist-003, Dist-005, Dist-007, Dist-010, Dist-013 and Dist-015, with mutations of 3%, 5%,7%, 10%, 13%, 15%, respectively[2]. Now using the mutated genomes, 5000 reads were generated using standard ReadSim simulator parameters with forward orientation.

$$ComputationCosts = O((\frac{N}{p})^2) + O((L_R)^2) + O((\frac{N}{p})L_R L_G) \quad (1)$$

$$CommunicationCosts = O(p^2 L_R) + O(p\log p) + O(\frac{N}{pL_R}) + O(NL_G \log p) \quad (2)$$

$$T \approx O((\frac{N}{p})^2) + O((L_R)^2) + O((\frac{N}{p})L_R L_G) + O(\frac{N}{pL_R}) + O(NL_G \log p) \quad (3)$$

The generated reads from these mutated genomes were then aligned with the wildtype using the proposed P-Pyro-Align Algorithm. This procedure is adopted because generally scientists only have a wildtype of the microbial genomes available and therefore it depicts a more practical scenario.

After the alignment, a majority consensus of the reads is obtained. A metric referred to as *quality score* is defined that is equal to the percentage of the correct consensus obtained from the alignment with respect to the true genome. The quality score is then calculated for

[2]Random mutations at distinct positions as well as inserting rows of mutations from 2bp to 17bp were used

the consensus obtained from the aligned reads with the original genome from which the reads were generated.

We compare the accuracy of the algorithm with two different methods. First we compare with the methods that use simple pair-wise alignment of the reads with the reference genome. Simple pairwise alignment is used to show how bad the alignment can be with respect to the original when multiple genomes are used. Second, we compare it with a gap propagation method, proposed in [2]. For this, we have used ShoRAH software package [3] which uses a pairwise gap propagation method to mimic multiple sequence alignment and is similar to mapping algorithm such as Bowtie, MAQ etc. Simply put, gap propagation method builds multiple alignment from pairwise alignments by 'propagating' the gaps. Propagation of gaps is accomplished for every position where at least one read has an inserted base, a gap is inserted in the reference genome and, consequently, in all reads that overlap the genome at that position. The complexity of the procedure is at least $O(N^2)$, and it overestimates the gaps. However, it mimics multiple alignment and thus is a good comparison with our system. A majority consensus is obtained from each type of alignments discussed above. Our experiments suggest that in the event of multiple genomes mapping algorithms such as Bowtie perform close to gap propagation method, if the genetic difference in the genomes is not large.

The results of the alignments obtained and the accuracy of the consensus are shown in Fig. 6 for 5000 reads. The accuracy of the consensus obtained using just the pairwise alignment is less than 58% for 3% mutations and continues to decrease with 42% accuracy for 15% mutations. The quality obtained from the P-Pyro-Align is always greater than 93% and exhibits a sustained quality with increasing mutations. The accuracy of the gap propagation procedure is comparable to P-Pyro-Align for small mutations, but as the mutations increase the accuracy of gap propagation decreases and is also not feasible for large number of reads.

To illustrate that the alignment system works with reads from multiple genomes, we have used a mixture of mutated reads from Dist-003, Dist-005, Dist-007, Dist-010, Dist-013 and Dist-015 to get a new set of reads. The new set contains reads (2500 from each genome) from the mutated sets of genomes. The reads are then aligned by the P-Pyro-Align algorithm using wildtype as the reference genome. The results of alignment for this mixture set are shown in Fig. 7 for Dist-003/Dist-005, Dist-005/Dist-007, Dist-007/Dist010, Dist-010/Dist-013 and Dist-013/Dist-015 mixtures. It must be noted here that we don't have a 'ground truth' genome in these cases and hence no genome is available to compare the consensus obtained from the alignment.

However, we do know the mutation rates for the genomes from which the mixture sets were generated. Therefore, if an optimal alignment of these reads is obtained, the 'mutation' in the consensus should not be greater than the combined mutations of the genomes. For example consider the case of Dist-003/Dist-005 mixture. We know the mutation rates for the genomes from which the reads generated were 3% and 5% with respect to the wildtype. Therefore, for accurate alignment, the consensus of the alignment should not vary more than 8%, in the worst case, when compared to the wildtype. Same would be true for the other cases considered according to the mutation rates of the genomes. As can be seen that the results of the alignment compared with the wildtype are well within the expected limits. The accuracy of the pairwise alignment of the reads with the reference genome (in this case the wildtype), and that obtained using propagation method is also shown for comparison.

The quality of the alignment produced using P-Pyro-Align also had to be tested for various other parameters such as number of reads and the length of the reads. We present the results for our algorithm with increasing average length of the reads and with increasing number of

reads. The quality of the alignment for 100*bp* to 400*bp* reads using P-Pyro-Align on 8 processors is shown in Figure 8. As can be seen from the figure that with increasing length of the read the quality of the alignment is always greater than 99.6%.

The quality of the alignment produced with increasing number of reads for upto 100, 000 is reported in Fig 9. As can be seen that with increasing number of reads, the quality does not deteriorate and is always greater than 99%. Some decrease in the quality can be attributed to the merge strategy that may introduce additional indels for large number of reads.

If the parallel alignment algorithm does not sustain quality with increasing number of processors, then the scalability of the system is limited. We have tested the quality produced by P-Pyro-Align with increasing number of reads upto 100, 000 and 24 processors. As can be seen in Fig 10 the alignment produced with increasing decomposition granularity has no significant effect on the quality. With increasing number of processors of upto 24 processors and 100, 000 reads is never less than 98%.

4.2 Performance in terms of High Performance Computing Parameters

In this section we analyze the performance of our algorithm in terms of traditional HPC parameters such as execution time, speedups and scalability. The objective of the evaluation is to determine the advantages of the proposed P-Pyro-Align algorithm in terms of speedups and reduction in execution time.

For the sake of coherence we generated similar kinds of sets for execution time evaluation as were selected for quality assessment. We report results for up to 0.5 million reads. To the best of authors knowledge, there are no published reports of multiple aligning of such large number of reads in the literature.

As can be seen in Fig. 11 that the execution time decreases sharply with increasing number of processors. We are able to align 100, 000 reads on a 24 processor system in just 139 minutes. The timing for aligning 200, 000 reads on a 24 processor system is 278 minutes. The same number of reads would take approximately 7 days on a single processor. The number of days to get alignment on single processor system increases drastically with increasing number of reads.

As shown in Fig. 12, P-Pyro-Align exhibits linear speedup on a 24 processor system. For less number of reads, the speedups initially obtained are near-linear e.g. for 20, 000 reads the speedup obtained for 8 processor nodes is approximately factor 7; but with increasing processors the speedup decreases. This is due to the fact that with smaller number of reads and more processors, the amount of communication overhead is comparatively more compared to the computation costs. However, it can be seen that with increasing number of reads, e.g. 0.5 million, super-linear speedups can be observed. This is because $(1/p)^2$ factor decrease becomes dominant as the number of reads on each processor increases, and it is in agreement with the complexity analysis in Section 3. It is safe to assume that the same behavior would be exhibited with increasing number of reads and processors.

5 Conclusion

With rapidly increasing knowledge of variants, alignment of reads from multiple genomes would be of increasing importance. We have demonstrated that multiple alignment for large number of error-prone reads is not only possible but it can also produce meaningful results. It gives access to regions that are highly divergent from the first reference or wildtype and would be rendered useless with mapping/pairwise alignments with the reference genome.

To this end, we have presented a highly scalable data parallel algorithm to multiple align large number of short reads from the pyrosequencing procedure. The proposed multiple alignment can take advantage of data decomposition to get results in a reasonable time. We also presented the quality assessment results and compared those with the results obtained by simple pair-wise alignment procedure and 'propagation' methods.

We emphasize here, as of today there are no *multiple alignment* methods currently available to align short error prone pyrosequencing reads upto 0.5 million with a potential to align even larger number of reads on large parallel systems. We believe that the parallel alignment method presented in the paper would find its uses in a multitude of computational biology and bioinformatics problems requiring alignments of large number of reads.

References

1. CAH. Dna sequencing: bench to bedside and beyond. Nucleic Acids Research. 2007; 35:62276237.
2. Eriksson N, Pachter L, Mitsuya Y, Rhee SY, Wang C, Gharizadeh B, Ronaghi M, Shafer RW, Beerenwinkel N. Viral population estimation using pyrosequencing. PLoS Comput Biol. May.2008 4:e1000074. [PubMed: 18437230]
3. Wang, Mitsuya; Gharizadeh, Ronaghi; Shafer. Characterization of mutation spectra with ultra-deep pyrosequencing: application to hiv-1 drug resistance. Genome Res. 2007 Aug; 17(8):1195–201. [PubMed: 17600086]
4. Liu Z, Lozupone C, Hamady M, Bushman FD, Knight R. Short pyrosequencing reads suffice for accurate microbial community analysis. Nucl Acids Res. 2007:gkm541.
5. Hou1 H-YJZCX-L, Cao Q-Y. Pyrosequencing analysis of the gyrb gene to differentiate bacteria responsible for diarrheal diseases. European Journal of Clinical Microbiology & Infectious Diseases. 2007; 27(7):587–596.
6. Campagna D, Albiero A, Bilardi A, Caniato E, Forcato C, Manavski S, Vitulo N, Valle G. Pass: a program to align short sequences. Bioinformatics. 2009; 25:967–968. [PubMed: 19218350]
7. Coarfa C, Milosavljevic A. Pash 2.0: scaleable sequence anchoring for next-generation sequencing technologies. Pac Symp Biocomput. 2008; 13:102–113. [PubMed: 18229679]
8. Eaves H, Gao Y. Mom: maximum oligonucleotide mapping. Bioinformatics. 2009; 25:969–970. [PubMed: 19228804]
9. Hormozdiari F, Alkan C, Eichler E, Sahinalp S. Combinatorial algorithms for structural variation detection in high-throughput sequenced genomes. Genome Res. 2009; 19:1270–1278. [PubMed: 19447966]
10. Jiang H, Wong W. Seqmap: mapping massive amount of oligonucleotides to the genome. Bioinformatics. 2008; 24:2395–2396. [PubMed: 18697769]
11. Langmead B, Trapnell C, Pop M, Salzberg S. Ultrafast and memory-efficient alignment of short dna sequences to the human genome. Genome Biol. 2009; 10:R25. [PubMed: 19261174]
12. Li H, Durbin R. Fast and accurate short read alignment with burrows-wheeler transform. Bioinformatics. 2009; 25:1754–1760. [PubMed: 19451168]
13. Li H, Ruan J, Durbin R. Mapping short dna sequencing reads and calling variants using mapping quality scores. Genome Res. 2008; 18:1851–1858. [PubMed: 18714091]
14. Li R, Li Y, Kristiansen K, Wang J. Soap: short oligonucleotide alignment program. Bioinformatics. 2008; 24:713–714. [PubMed: 18227114]
15. Li R, Yu C, Li Y, Lam T, Yiu S, Kristiansen K, Wang J. Soap2: an improved ultrafast tool for short read alignment. Bioinformatics. 2009; 25:1966–1967. [PubMed: 19497933]
16. Malhis N, Butterfield Y, Ester M, Jones S. Slider: maximum use of probability information for alignment of short sequence reads and snp detection. Bioinformatics. 2009; 25:6–13. [PubMed: 18974170]
17. Ning Z, Cox A, Mullikin J. Ssaha: a fast search method for large dna databases. Genome Res. 2001; 11:1725–1729. [PubMed: 11591649]

18. Prufer K, Stenzel U, Dannemann M, Green R, Lachmann M, Kelso J. Patman: rapid alignment of short sequences to large databases. Bioinformatics. 2008; 24:1530–1531. [PubMed: 18467344]
19. Rumble S, Lacroute P, Dalca A, Fiume M, Sidow A, Brudno M. Shrimp: accurate mapping of short color-space reads. PLoS Comput Biol. 2009; 5:e1000386. [PubMed: 19461883]
20. Schatz M. Cloudburst: highly sensitive read mapping with mapreduce. Bioinformatics. 2009; 25:1363–1369. [PubMed: 19357099]
21. Smith A, Xuan Z, Zhang M. Using quality scores and longer reads improves accuracy of solexa read mapping. BMC Bioinformatics. 2008; 9:128. [PubMed: 18307793]
22. Weese D, Emde A, Rausch T, Doring A, Reinert K. Razers: fast read mapping with sensitivity control. Genome Res. 2009; 19:1646–1654. [PubMed: 19592482]
23. Alkan C, Kidd J, Marques-Bonet T, Aksay G, Antonacci F, Hormozdiari F, Kitzman J, Baker C, Malig M, Mutlu O, Sahinalp S, Gibbs R, Eichler E. Personalized copy number and segmental duplication maps using next-generation sequencing. Nat Genet. 2009
24. Lin H, Zhang Z, Zhang M, Ma B, Li M. Zoom! zillions of oligos mapped. Bioinformatics. 2008; 24:2431–2437. [PubMed: 18684737]
25. Langmead B, Trapnell C, Pop M, Salzberg S. Ultrafast and memory-efficient alignment of short dna sequences to the human genome. Genome Biol. 2009; 10:R25. [PubMed: 19261174]
26. Schneeberger K, Hagmann J, Ossowski S, Warthmann N, Gesing S, Kohlbacher O, Weigel D. Simultaneous alignment of short reads against multiple genomes. Genome Biology. 2009; 10(9):R98. [PubMed: 19761611]
27. Clark R, Schweikert G, Toomajian C, Ossowski S, Zeller G, Shinn P, Warthmann N, Hu T, Fu G, Hinds D, Chen H, Frazer K, Huson D, Scholkopf B, Nordborg M, Ratsch G, Ecker J, Weigel D. Common sequence polymorphisms shaping genetic diversity in arabidopsis thaliana. Science. 2007; 317:338–342. [PubMed: 17641193]
28. Zeller G, Clark R, Schneeberger K, Bohlen A, Weigel D, Ratsch G. Detecting polymorphic regions in arabidopsis thaliana with resequencing microarrays. Genome Res. 2008; 18:918–929. [PubMed: 18323538]
29. Ossowski S, Schneeberger K, Clark R, Lanz C, Warthmann N, Weigel D. Sequencing of natural strains of arabidopsis thaliana with short reads. Genome Res. 2008; 18:2024–2033. [PubMed: 18818371]
30. Wang L, Jiang T. On the complexity of multiple sequence alignment. Journal of Computational Biology. 1994; 1(4):337–348. [PubMed: 8790475]
31. Notredame C, Higgins D, Heringa J. T-coffee: A novel method for multiple sequence alignments. Journal of Molecular Biology. 2000; 302:205–217. [PubMed: 10964570]
32. Edgar R. Muscle: Multiple sequence alignment with high accuracy and high throughput. Nucleic Acids Research. 2004; 32(5)
33. Thompson J, Higgins D, Gibson TJ. Clustal w: improving the sensitivity of progressive multiple alignment through sequence weighting, position-specific gap penalties and weight matrix choice. Nucleic Acids Res. 1994; 222:4673–4690. [PubMed: 7984417]
34. Do C, Mahabhashyam M, Brudno M, Batzoglou S. Probcons: Probabilistic consistency-based multiple sequence alignment. Genome Research. 2005; 15:330–340. [PubMed: 15687296]
35. Morgenstern B. Dialign: multiple dna and protein sequence alignment at bibiserv. Nucleic Acids Research. 2004; 32:W33–W36. [PubMed: 15215344]
36. Saeed, F.; Khokhar, A.; Zagordi, O.; Beerenwinkel, N. Multiple sequence alignment system for pyrosequencing reads. Proceedings of the 1st International Conference on Bioinformatics and Computational Biology(BICoB); 2009. p. 362-375.
37. Saeed F, Khokhar AA. A domain decomposition strategy for alignment of multiple biological sequences on multiprocessor platforms. J Parallel Distrib Comput. 2009; 69(7):666–677.
38. Saeed, F.; Khokhar, A. Sample-align-d: A high performance multiple sequence alignment system using phylogenetic sampling and domain decomposition. Proc. 23rd IEEE International Parallel and Distributed Processing Symposium; April 2007;
39. Schmid, MSR.; Schuster, SC.; Huson, D. Readsim- a simulator for sanger and 454 sequencing. 2006.

40. Hanmao S, Jonathan S. Parallel Sorting by Regular Sampling. Journal of Parallel and Distributed Computing. 1992; 14(4):361–372.
41. Needleman SB, Wunsch CD. A general method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of Molecular Biology. Mar.1970 48:443–453. [PubMed: 5420325]
42. Setubal, C.; Meidanis, J. Introduction to Computational Molecular Biology. PWS Publishing; Jan. 1997
43. Katoh K, Misawa K, Kuma K, Miyata T. Mafft: A novel method for rapid multiple sequence alignment based on fast fourier transform. Nucleic Acids Res. 2002; 30(14):3059–3066. [PubMed: 12136088]
44. Edgar R. Muscle: A multiple sequence alignment method with reduced time and space complexity. BMC Bioinformatics. 2004:1471–2105.
45. Jones DT, Taylor WR, Thornton JM. The rapid generation of mutation data matrices from protein sequences. BMC Bioinformatics. 1991; 8(3):275–282.
46. Mller T, Spang R, Vingron M. Estimating amino acid substitution models: A comparison of dayhoff's estimator, the resolvent approach and a maximum likelihood method. Mol Bio Evol. 2002; 19(1):8–13. [PubMed: 11752185]
47. Edgar RC, Sjolander K. A comparison of scoring functions for protein sequence profile alignment. Bioinformatics. 2004; 20(8):1301–1308. [PubMed: 14962936]
48. Hambrusch SE, Khokhar A. C^3: A parallel model for coarse-grained machines. Journal of Parallel and Distributed Computing. 1996; 32(2):139–154.
49. Hambrusch SE, Hameed F, Khokhar AA. Communication operations on coarse-grained mesh architectures. Parallel Computing. 1995; 21(5):731–751.
50. Kumar, V.; Grama, A.; Gupta, A.; Karpis, G. Introduction to Parallel Computing: Design and Analysis of Parallel Algorithms. Benjamin-Cummings Pub Co; Jan. 1994
51. Huang, JS.; Chow, YC. Parallel sorting and data partitioning by sampling. Proceedings of the IEEE Computer Society's Seventh International Computer Software and Applications Conference; 1983. p. 627-631.
52. Gusfield, D. Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology. Cambridge University Press; Jan. 1997
53. Thompson JD, Plewniak F, Poch O. Balibase: a benchmark alignment database for the evaluation of multiple alignment programs. Bioinformatics. 1999; 15(1):87–88. [PubMed: 10068696]

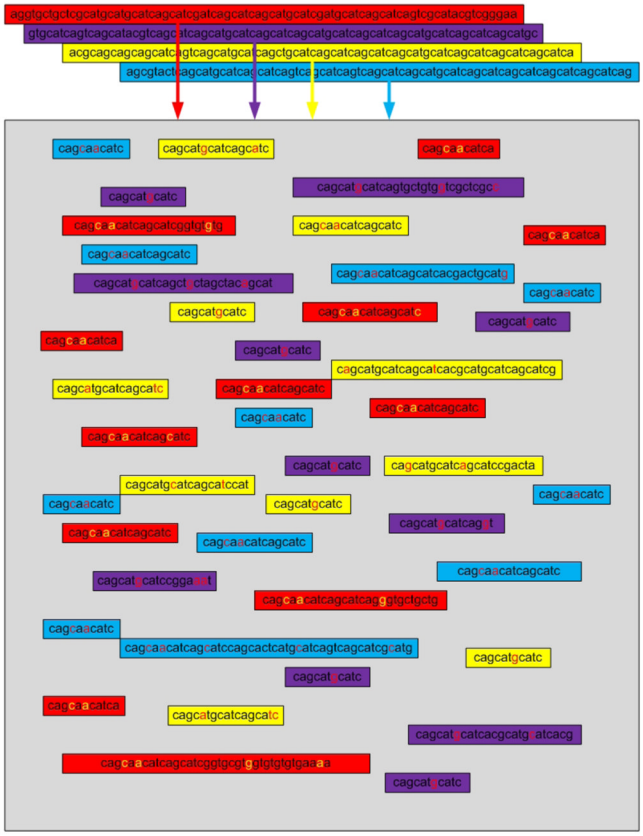

Figure 1.
Reads from four different divergent strains of genomes of the same species

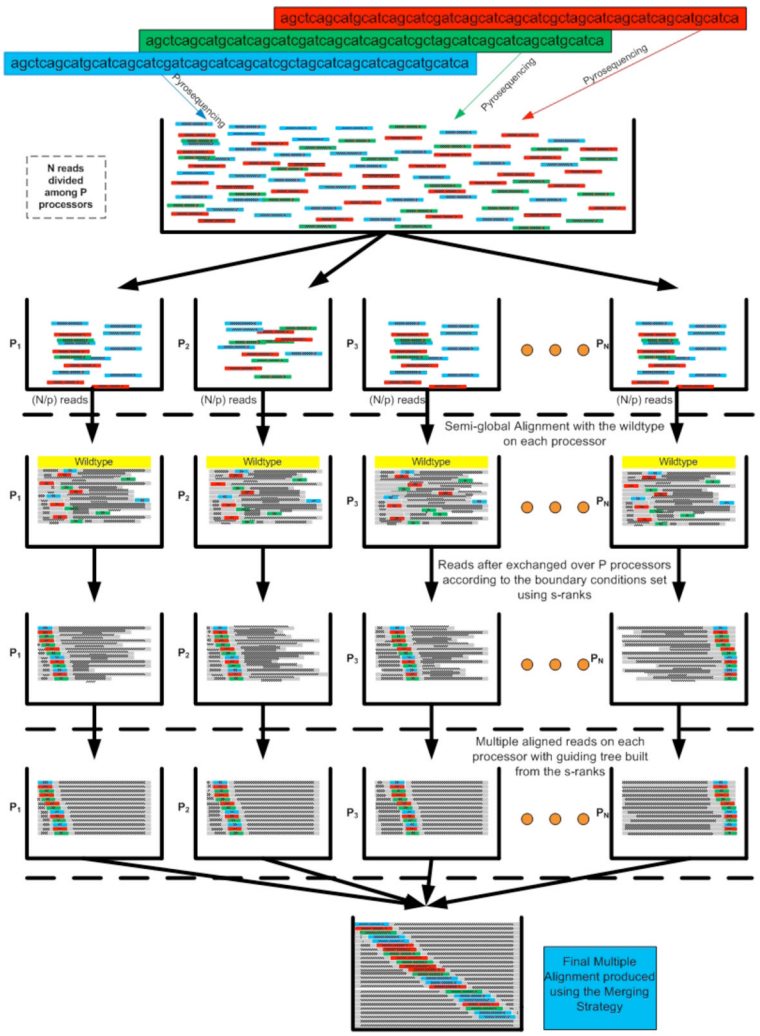

Figure 2.
A visually represented summary of the proposed algorithm

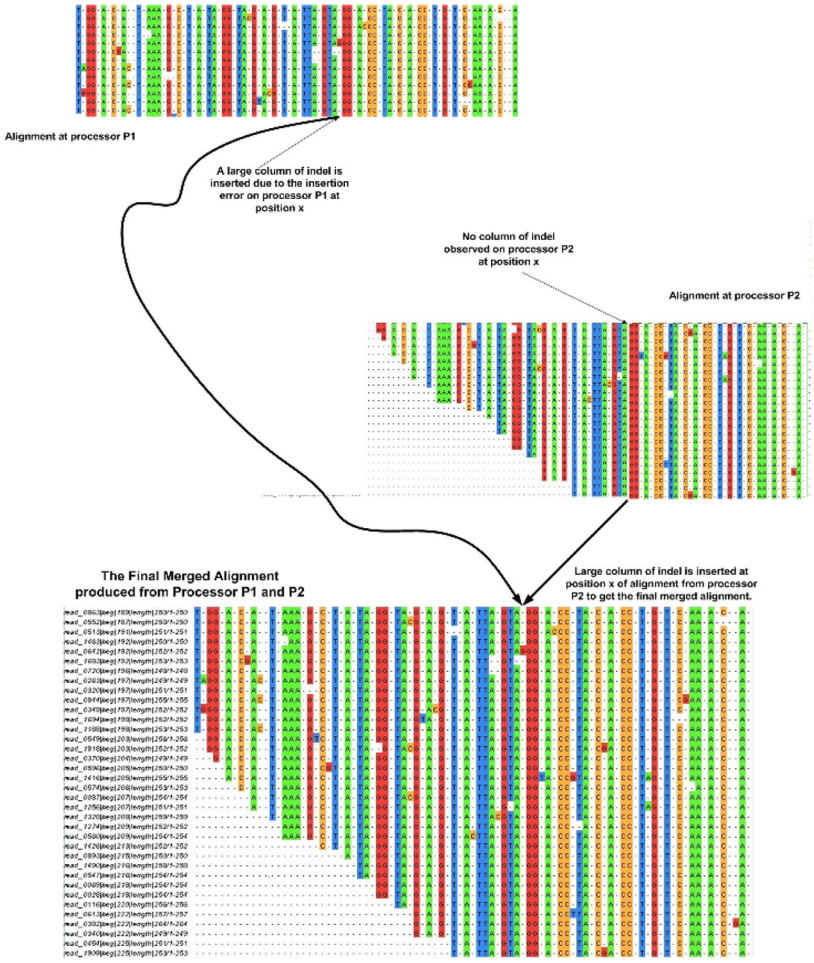

Figure 3.
The Merging strategy for alignments

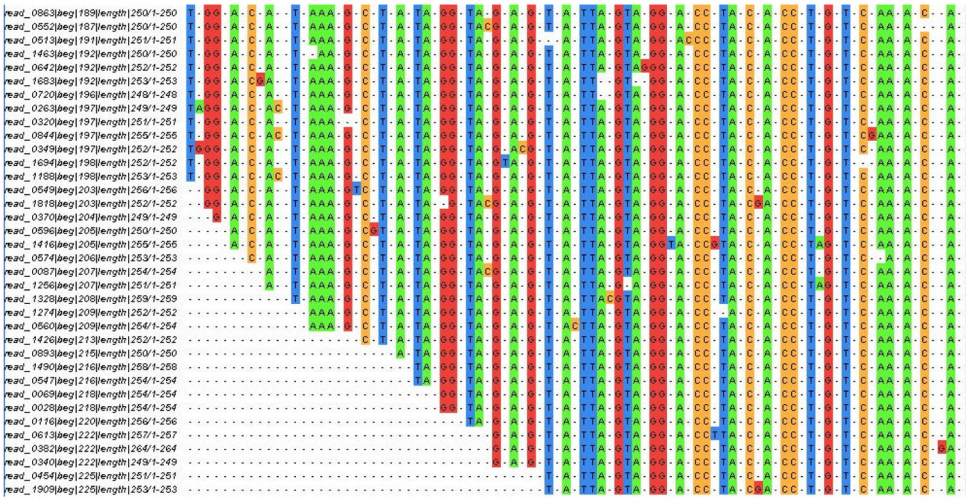

Figure 4.
The final Alignment of the reads

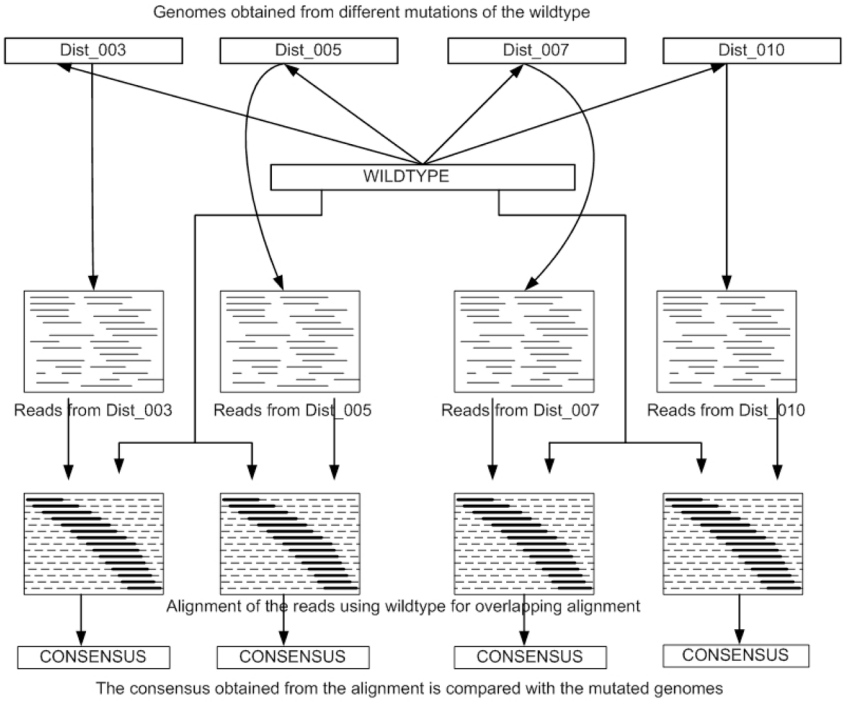

Figure 5.
The experimental setup for the quality assessment of the multiple alignment program

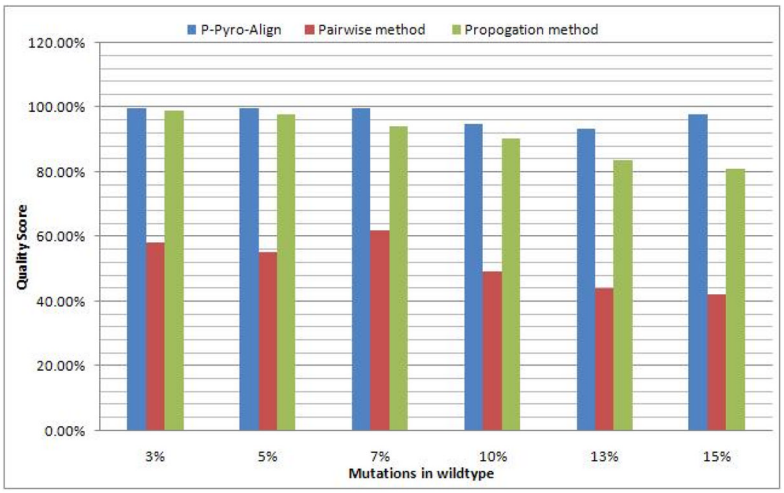

Figure 6.
The quality of alignment with different mutation rates for P-Pyro-Align, pairwise and gap propagation methods

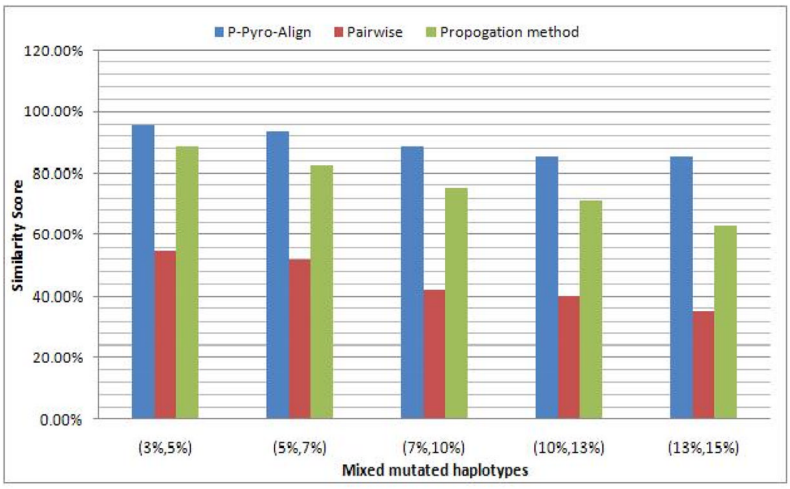

Figure 7.
The quality of alignment with different mixture of mutation rates for P-Pyro-Align, pairwise and gap propagation methods

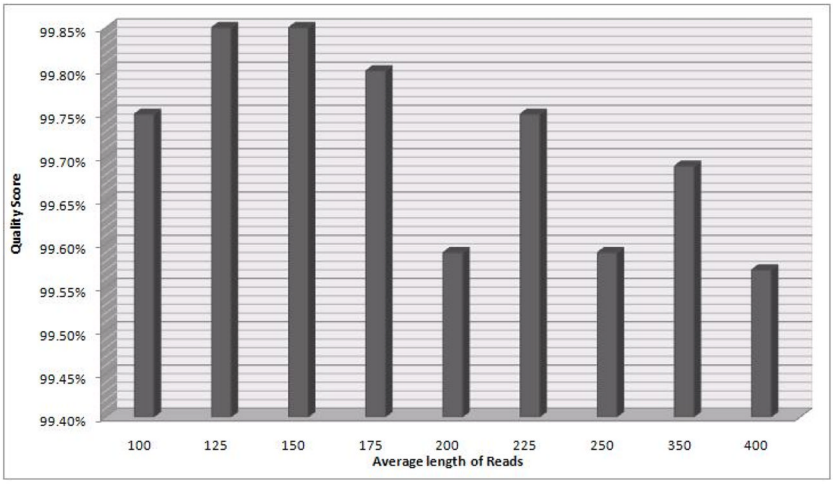

Figure 8.
The quality of alignment produced using P-Pyro-Align with increasing average length of the reads

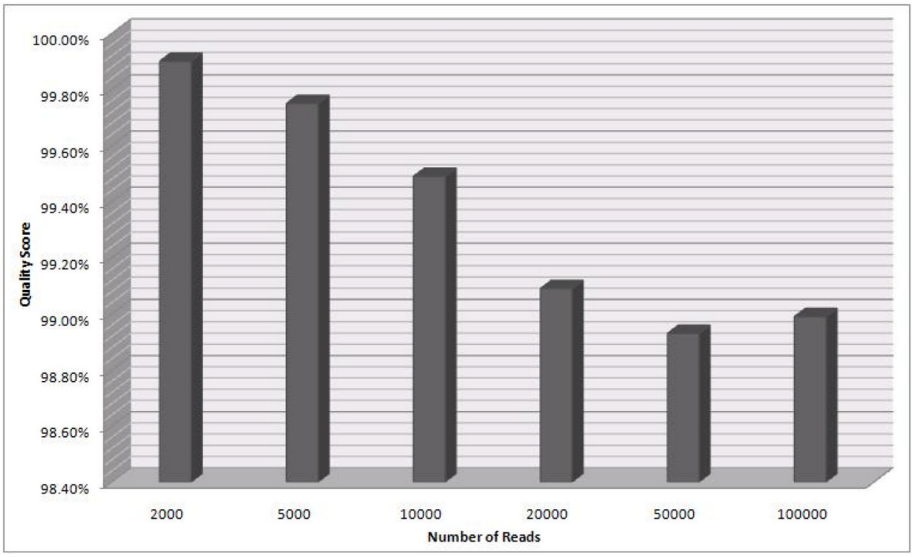

Figure 9.
The quality of alignment produced using P-Pyro-Align with increasing number of reads

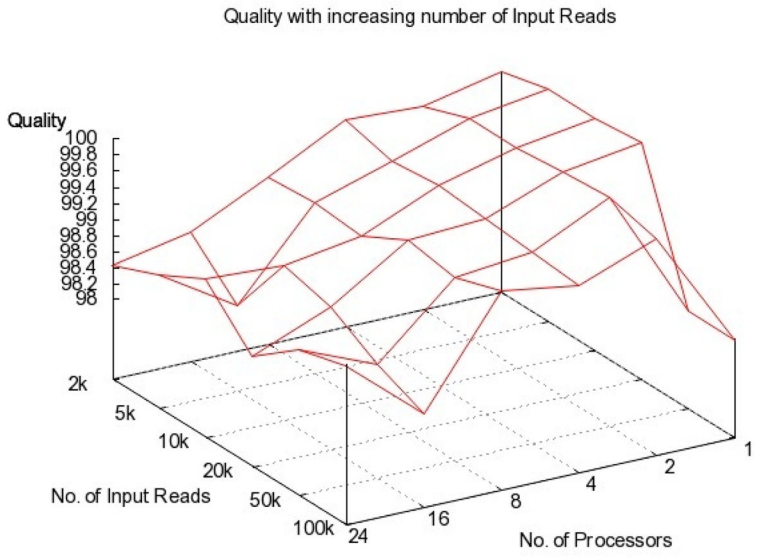

Figure 10.
The quality of alignment produced using P-Pyro-Align with increasing number of processors

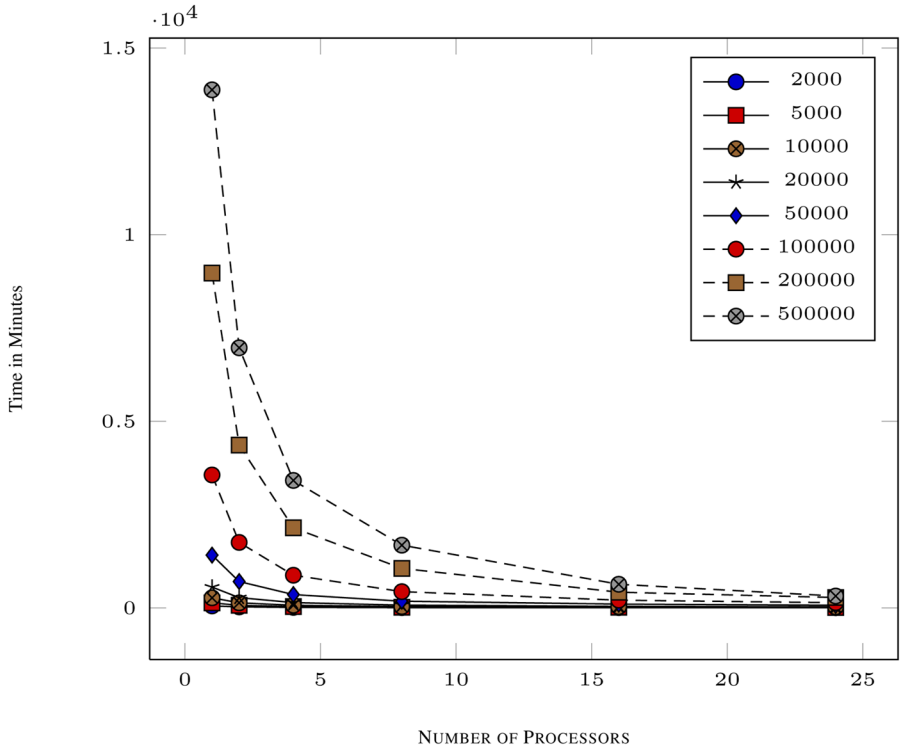

Figure 11.
Scalability of the execution time w.r.t. the number of processors

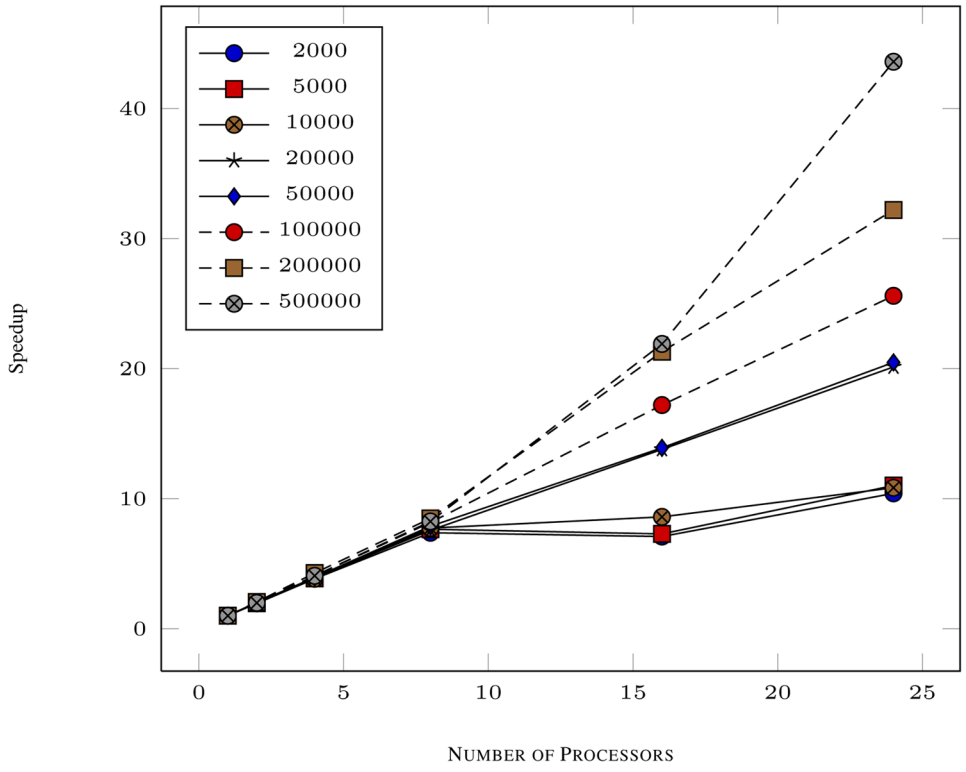

Figure 12.
Speedup for P-Pyro-Align with increasing number of processors

Table 1

Computation Costs:

STEP	O(Time)
Semi-global alignment of ($w = N/p$) reads with reference genome	$w \times LR \times LG$
Sorting of N/p sequences based on s-rank	$w \log w$
Sample $k = p - 1$ sequences	w
Sorting of $k \times p$ sample s-ranks	$(k \times p)log(k \times p)$
Clustalw executed on ($w = N/p$) sequences in parallel	$w^2 + L_R 2$
Computations for merging	$2 \times w \times LG$
TOTAL Computation Cost (for $w=\frac{N}{p}$)	$O((\frac{N}{p})^2+(L_R)^2+(\frac{N}{p})L_R L_G)$

Algorithm 1

P-Pyro-Align

Require: p processor for computation
Require: N pyrosequencing reads and reference-genome/wildtype, such that a distinct block of N/p reads is assigned to each processor.
Ensure: Gaps are inserted in each of the reads so that

- All reads have the same length (including trailing and leading gaps)
- Score of the global map is maximum and the error insertions and deletions are 'highlighted' after the alignment.

1 Locally compute a semi-global alignment of all the sequences with the reference genome in each processor (This will place each read in the correct position w.r.t genome).

2 Use parallel Sample sort to sort the reads based on s-ranks

3 Align sequences in each processor using progressive alignment system with trailing and leading gaps. The gaps are assigned different weights based on the position of the reads. The leading and trailing gaps are assigned weights different than the indels that are encountered in between base pairs of the reads.

4 Merge the aligned reads on each processor using a customized merging tree.

5 Continue the merging till all sequences are merged to produce a global multiple alignment.

6 Output the final alignment.