2014 22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing

A Parallel Algorithm for the Best k-mismatches Alignment Problem

Cristian Del Fabbro
Institute of Applied Genomics
Udine, Italy
Email: delfabbro@appliedgenomics.org

Fabio Tardivo
University of Udine
Udine, Italy
Email: fabio.tardivo@spes.uniud.it

Alberto Policriti
University of Udine and
Istitute of Applied Genomics
Udine, Italy
Email: alberto.policriti@uniud.it

***Abstract*—We propose a parallel algorithm that solves the *best k-mismatches alignment problem* against a genomic reference using the "one sequence/multiple processes" paradigm and distributed memory. Our proposal is designed to take advantage of a computing cluster using MPI (Message Passing Interface) for communication. Our solution distributes the reference among different nodes and each sequence is processed concurrently by different nodes. When a (putative) best solution is found, the successful process propagates the information to other nodes, reducing search space and saving computation time.**

The distributed algorithm was developed in C++ and optimized for the PLX and FERMI supercomputers, but it is compatible with every OpenMPI-based cluster. It was included in the ERNE (Extended Randomized Numerical alignEr) package, whose aim is to provide an all-inclusive set of tools for short reads alignment and cleaning. ERNE is free software, distributed under the Open Source License (GPL V3) and can be downloaded at: `http://erne.sourceforge.net`. The algorithm described in this work is implemented in the ERNE-PMAP and ERNE-PBS5 programs, the former designed to align DNA and RNA sequences, while the latter is optimized for bisulphite-treated sequences.

I. Introduction

The advent of NGS (Next Generation Sequencing), first appeared in 2005, has changed the bioinformatics field, opening new and unimaginable research perspectives. New sequencers are able to produce huge amounts of data, at a very low cost and in a few days. The sequencers produce a set of short sequences (called "reads") in the alphabet $\{A, C, G, T, N\}$. The first four letters represent nucleotide bases that can be present in a genome (Adenine, Cytosine, Guanine and Thymine). Since the sequencing reading process is not perfect, in some cases the sequencer prefers to return a "not known" signal (N) instead of returning an incorrect value.

In bioinformatics, the *short string alignment* problem is the problem of aligning (searching) the "correct" position for each short read against a reference (a representation of a genome similar but not equal to the sequenced individual), allowing only a limited amount of mismatches. In some cases one wants also to allow a (still more) limited number of insertions and deletions of characters (bases) in the string. Often, the aligners use heuristics to cut the search space and hence reduce computation time, at the cost of a (hopefully) negligible amount of false positives and false negatives. There are numerous NGS aligners proposed by the scientific community, for a review see [1], [2].

The NGS sequencer technology has improved, since 2005, at a very fast rate: every year the throughput of the sequencers increased by a 5-fold factor [3], [4]. Such of high rate of data production imposes the need to reduce the time required to perform the alignment phase (the bottleneck in any resequencing or otherwise analysing project) without sacrificing accuracy. These trends in growth pose new computational challenges: the higher the amount of data to process, the higher the need to process this data as quickly as possible.

In this work we propose to use a computing cluster and partition both the set of reads and the reference genome across the nodes. The first ingredient is to use MPI (Message Passing Interface) [5] to transmit input and output of the alignments performed. We explored the approach consisting in allowing communication among nodes *during* the alignment phases. When a node finds a better solution than the ones currently discovered, the possibility to broadcast reduces search space and computation. This approach is particularly well-suited when variations of the so called "best" k-mismatch problem are under study. More on this aspect below.

A problem that can arise, when adding interprocess communications, could be the overhead caused by the communication itself: significant amounts of computation time spent in transmitting and/or waiting for data. Aware of this problem, we designed the communication system controlled to avoid flooding the transmission media and trying to keep delays to a minimum in data waiting.

The approach is based on an evolution of the mrNA software [6] and it was optimized to work on the PLX supercomputer [7]. We are planning to optimize the code for the Fermi supercomputer (12th in the world, [8]). However, the current implementation is compatible with every cluster supporting OpenMPI. In this paper we explore only the capability of the ERNE-PBS5 software (the parallel version of ERNE-BS5 [9]). ERNE-PBS5 is able to align reads produced using protocols for bisulfite treated reads (a protocol called BS-seq), that it is able to detect (un)methylated cytosines that are crucial information used in epigenetic studies. The BS-seq protocol transforms the majority of the cytosines into thymines, *reducing the sequence complexity and increasing the search space*, hence a set of reads produced using BS-seq in general requires more time to be aligned w.r.t. DNA or RNA reads. A description of the problems arising when working on BS-seq reads is out of the scope of this paper, for which we refer the reader to [2], [9]. Our implementation takes as input a reference and a set of reads in FASTQ format and produces a (standard in bioinformatics) SAM/BAM formated output.

1066-6192/14 $31.00 © 2014 IEEE
DOI 10.1109/PDP.2014.51

A. Parallel alignment approaches

To date, there are two main approaches/technologies employed in parallel processing. The first one is based on Message Passing Interface (MPI) and exist different "flavors", *e.g.*, OpenMPI [10] or Intel® MPI. Open source NGS alignment software based on MPI include mrNA [6], GNUMAP [11] and pBWA [12]. There exists also a proprietary solution called Novoalign.

The second approach is a framework proposed by Google and based on a technique called MapReduce [13] that can efficiently use from a few to thousands of (possibly rentable) machines. Different realizations are available, based on MapReduce or on Apache Hadoop (Open Source). A naive application to the alignment problem starts by dividing the reads into n groups. The computation (MAP) can now work independently on each group against the whole reference. The output (REDUCE) is then obtained simply as a concatenation of the output of each process. In the last few years different solutions based on the MapReduce paradigm appeared, *e.g.*, Crossbow [14], CloudBurst [15], SeqMapReduce [16], CloudAligner [17], and SEAL [18].

B. The best k-mismatches alignment problem

Let P (pattern/read) and R (reference) be two strings in the nucleotide alphabet $\Sigma = \{A, C, T, G, N\}$. We indicate a sub-string of P (or R) starting at position i and ending at position j as $P[i, j]$, with $i, j \in \mathbb{N}$, $0 \leq i \leq j < |P|$. We write $P[i]$ as a shortcut for $P[i, i]$. When aligning we must take care of sequencing errors (due to an incorrect acquisition of data) and to differences between the individual sequenced and the reference. Due to these limitations we can not use "exact" string matching but we must consider (some form of) "approximate" string matching. It is customary to introduce a *distance* function to keep errors under a threshold. The simplest and most used distance in bioinformatics is the *Hamming distance* [19]:

$$d_H(X, Y) = |\{i \mid X[i] \neq Y[i], 0 \leq i < |X|\}|, \tag{1}$$

that is defined when $X, Y \in \Sigma^*$ and $|X| = |Y|$. The Hamming distance counts the number of differences (*i.e.*, mismatches) between two strings. However, in the NGS computing area, "aligning" means to solve the *best k-mismatches alignment problem*: given a read P, a reference R, and $k \in \mathbb{N}$, determine all the position in R where P can be aligned with k' mismatches ($0 \leq k' \leq k$) and there exist no positions where P can be aligned with less than k' mismatches. In other words, the problem is to find (if existing) the m positions $I = \{i_1, \ldots, i_m\}$ where

$$d_H(P, R[i_1, i_1 + |P| - 1]) = \ldots = d_H(P, R[i_m, i_m + |P| - 1]) = k' \tag{2}$$

and for no position j, $d_H(P, R[j, j + |P| - 1]) < k'$.

The naive solution is to compare every base of R against every base of P, with a computation cost of $\mathcal{O}(|P| \cdot |R|)$. Due to the dimension of the reference (from few thousand to billions of base pairs), this is not a practicable solution. Many algorithms have been proposed to pre-process the reference and achieve computation cost proportional to the logarithm of reference's dimension (or better) and linear in pattern's size. The most used solutions are based on suffix trees/arrays [20] and the Burrows-Wheeler transform [21]. The alignment core of the ERNE programs [22], [23] is based on a modification of the Karp-Rabin algorithm [24] and the use of the *Pigeonhole principle*.

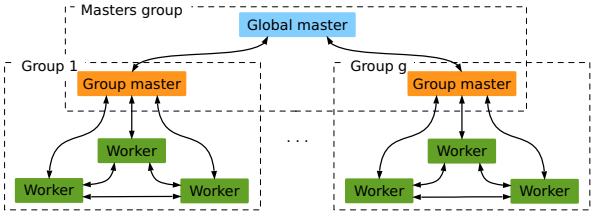

Fig. 1. Scheme of our implementation of Master/Workers pattern.

II. Implementation

All current cluster-oriented alignment tools use geometric decomposition to solve the best k-mismatches problem, because all inexact-matching algorithms proposed are strictly sequential. Therefore the only way to speed up the computation is to decrease the working load (*i.e.*, reducing CPU time). Our idea is not focused on a naive split of input data but on a runtime reduction of search space: if a process finds one or more occurrences with $k' < k$ mismatches, then all other processes must be notified in order to search for occurrences with k' mismatches instead of k, thereby reducing search space.

We chose the Master/Workers pattern to implement our algorithm, where workers can also communicate among them. This model allows both to apply geometric decomposition as well as sharing the "best" k-values among nodes. Usually biological interesting references (*e.g.*, human, mouse, grapevine) do not allow to splitting the reference in more than 10–30 chunks (chromosomes). Therefore the Master/Workers structure is replicated to fill all the nodes assigned to the job (*e.g.*, if 32 nodes are available and the reference is split into 8 pieces, then the structure will be replicated $32/8 = 4$ times). In this way each group (*i.e.*, the nodes whose union makes the reference) aligns a subset of input reads. In Fig. 1 is depicted an overview of the logical structure.

The structure is partitioned into two kinds of groups: the workers groups, composed of nodes that perform alignments, and the masters groups, containing the nodes whose tasks are data distribution and results collection. Let n be the number of nodes in the MPI environment and let m be the group size (*i.e.*, the numbers of parts in which the reference is split). The above groups are obtained partitioning the whole environment (`MPI_COMM_WORLD`) in $g = (n/m)$ workers groups (`workerIntracomm`) plus a single masters group (`mastersIntracomm`) consisting in group masters and the global master. This allows optimal data distribution and results collection. In particular, the global master can assign a block of reads to each worker group simply by sending it by `mastersIntracomm` to relative group master. Then the group master broadcasts the reads in the associated working group, while the global master can continue the assignment to another group. In a similar way results are collected: during the alignment phase results are stored in group master's local storage, so that when the phase is completed they can be easily

TABLE I. TABLE OF READS/SEC.

Groups	Nodes	*without errors update*			*with errors update*		
		Min	Average	Max	Min	Average	Max
1	1	18,16	22,70	30,68	n.a.	n.a.	n.a.
1	2	17,34	20,39	25,10	15,02	23,70	27,65
1	4	17,11	24,75	35,54	16,76	27,57	29,57
1	8	20,01	29,28	41,71	22,56	31,95	47,50
2	1	19,83	22,11	51,26	21,28	23,10	38,61
2	2	22,35	23,68	32,97	23,40	25,68	32,11
2	4	28,24	29,63	38,99	25,58	31,84	33,41
2	8	26,07	33,28	50,57	28,49	35,18	45,94
4	1	17,93	20,52	37,50	21,54	24,03	72,41
4	2	24,49	25,47	36,21	19,27	28,27	34,74
4	4	26,48	29,95	55,26	25,53	32,55	52,50
4	8	24,56	33,11	56,34	30,67	37,11	47,66

sent to global master using `mastersIntracomm`. This last step is necessary in order to produce a unique output file. For the sake of performance group masters are implemented as threads in nodes with rank equal to 0 according to relative `workersIntracomm`. The group master of workers group number 0 is also a global master (in other words, the global master and group masters are also workers). Other available cores in node's CPU are exploited by alignment threads. In order to reduce waste of time and to avoid communications bottleneck problems, threads use blocking mechanisms to send/receive data, hence most of the processors time is used for alignments. Usually input data is huge, hence the sequence load-distribute-align-collect-save is looped until input is consumed.

The most delicate part of our implementation is the update of best k-values. The time at which this action is performed is unpredictable and updates are from unknown sources, so the primitives involved must be poll-able. These two conditions can not be met by `MPI_Bcast` procedure because they can not be detected by the standard MPI probing mechanisms (*e.g.*, `MPI_Probe`). In order to solve this issue we implemented a poll-able broadcast [25] that uses point-to-point communications on a binomial tree virtual topology. In a group with m nodes it needs $log_2(m)$ propagation steps and is compatible with `MPI_Probe`, hence it can be executed only if necessary. Also an efficient buffer mechanism is implemented to avoiding waiting network operations. It consists of a thread that waits for communications and update each process's k values. In this fashion the alignment threads can simply check if and how k has changed, without wasting time in communication checks.

As a general (practical) consideration on our approach, consider the fact that when found, the best alignment allows to cut search space to all processes reached by the message broadcasted. Even though cannot claim a better worst-case performance, this consideration had practical value on our experiments.

III. RESULTS

We tested our implementation using *Vitis vinifera* genome as reference and a set of 161,847,352 BS-seq reads, 100 base pairs long. The reference genome is composed of 19 chromosomes and we generated four different partitions for the groups. The partitions are optimized for 1, 2, 4 and 8 nodes.

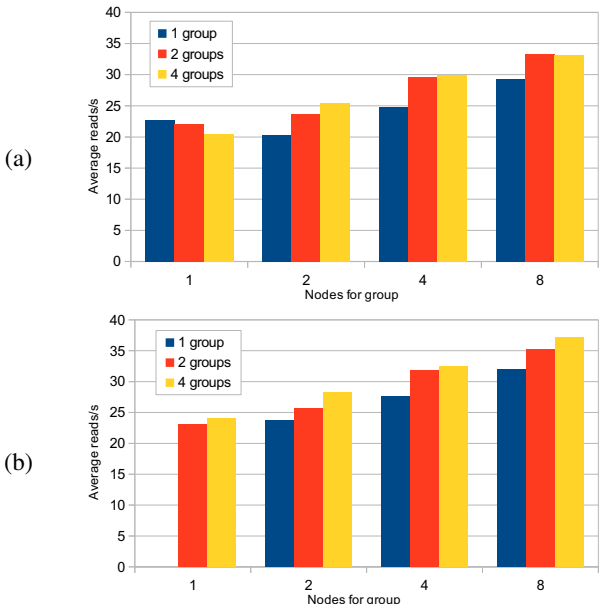

Fig. 2. Processivity of the implementation *without* (a) and *with* (b) errors update communications. On Y-axis is the *average* number of aligned reads per second that a node is able to process. On X-axis is the number of nodes for group.

Tests are performed on PLX supercomputer. This cluster is composed by 254 nodes, each one equipped with two esa-core CPUs at 2.4 GHz (12 total cores) and 48 GB of DDR3 RAM at 1,333 MHz. The network is entrusted to a Infiniband connection with 4x QDR switches and the operating system is Red Hat RHEL 5.6 x86_64.

We run different alignments with 1, 2 and 4 groups and with 1, 2, 4 and 8 nodes for each group. We collected the reads processed per second during the alignments phase and we summarized the results showing minimum, average, and maximum reads/sec in Table I. We tested the algorithm *with* and *without* the errors update procedure. The "without" tests represent a MapReduce model only, while the "with" tests represent the complete ERNE-PBS5 model (MapReduce and errors broadcast and update). The average columns are graphically depicted in Fig. 2. The case "one group with one node" was not run using ERNE-PBS5, which require at least two nodes, and the non-parallel version ERNE-BS5 was used instead. In this case there are no communication and so we reported the results only on the "without" section. We can argue that the number of reads per second that a node can process increases (roughly) linearly with the number of nodes in the group. This is possible only due to the communication system we have implemented. Since the groups involved in an alignment works on different set of reads, using two or more groups does not improve performances on a node (as expected). Our current implementation is not optimized for data communication, hence the measurements are taken only on the alignment phase. Our next goal is to improve on these bottlenecks and to enhance the whole program for faster

alignment.

We tested the ability of working with different groups because we plan to optimize the code for the Fermi [8] infrastructure, where the minimum allocation unit for a parallel program is 64 nodes (up to 2048 nodes). In this situation we were not able to divide the grapevine genome in 64 parts, but, due to our implementation, we were allowed to use 8 groups of 8 nodes (up to 256 groups). This allows us to use a MapReduce-like approach partitioning reads into groups, maintaining a partition of the reference and allowing communication within each group.

IV. Conclusion

In this work we explored state-of-the-art parallel tools for the alignment problem. We started from the two most popular family of solutions: ad-hoc system that use naive messages paradigm and the MapReduce-like approaches. All analyzed tools have good features but none implements a powerful ad-hoc system combined with MapReduce idea. So we designed and implemented a comprehensive software that uses MapReduce-like decompositions of the reference on top of a novel view on parallel alignments. This approach consist in a Master/Workers architecture where workers can share the results during alignment phase in order to reduce search space. The reference's biological meaning limits the number of parts in which the reference's data-structure can be divided. To solve this issue we allowed to partition the reference into a group of nodes with a Master/Workers structure. Then the set-up is automatically replicated to fill available resources so that each copy aligns a subset of input reads.

We tested our algorithm with real data and we reached our goal: communication among processes guarantees a reduction of CPU time, allowing each node to process more reads per second (w.r.t. to the serial or MapReduce only implementations).

As a further improvement, we plan to replace the legacy Boost Thread Library with the more performing and portable OpenMP APIs [26] and enable it to GPGPU (General-Purpose computing on Graphics Processing Units) by OpenCL or CUDA as done in [27].

Acknowledgment

Funding: This work was partially supported by Epigenomics Flagship Project (Progetto Bandiera Epigenomica), MIUR-CNR.

References

[1] H. Li and N. Homer, "A survey of sequence alignment algorithms for next-generation sequencing," *Briefings in bioinformatics*, vol. 11, no. 5, pp. 473–483, 2010.

[2] C. Bock, "Analysing and interpreting DNA methylation data," *Nature Reviews Genetics*, vol. 13, no. 10, pp. 705–719, 2012.

[3] M. L. Metzker, "Sequencing technologies - the next generation," *Nature Reviews Genetics*, vol. 11, no. 1, pp. 31–46, 2009.

[4] M. C. Schatz, B. Langmead, and S. L. Salzberg, "Cloud computing and the DNA data race," *Nature biotechnology*, vol. 28, no. 7, p. 691, 2010.

[5] MPI Forum, "MPI: A Message-Passing Interface Standard. Version 2.2," September 4th 2009, available at: http://www.mpi-forum.org (Dec. 2009).

[6] C. Del Fabbro, F. Vezzi, and A. Policriti, "mrNA: the MPI randomized Numerical Aligner," in *Bioinformatics and Biomedicine (BIBM), 2011 IEEE International Conference on*. IEEE, 2011, pp. 139–142.

[7] [Online]. Available: http://www.top500.org/system/177417

[8] [Online]. Available: http://www.top500.org/system/177720

[9] N. Prezza, C. Del Fabbro, F. Vezzi, E. De Paoli, and A. Policriti, "ERNE-BS5: aligning BS-treated sequences by multiple hits on a 5-letters alphabet," in *Proceedings of the ACM Conference on Bioinformatics, Computational Biology and Biomedicine*. ACM, 2012, pp. 12–19.

[10] E. Gabriel, G. E. Fagg, G. Bosilca, T. Angskun, J. J. Dongarra, J. M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine *et al.*, "Open MPI: Goals, concept, and design of a next generation MPI implementation," in *Recent Advances in Parallel Virtual Machine and Message Passing Interface*. Springer, 2004, pp. 97–104.

[11] N. L. Clement, M. J. Clement, Q. Snell, and W. E. Johnson, "Parallel mapping approaches for GNUMAP," in *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on*. IEEE, 2011, pp. 435–443.

[12] D. Peters, X. Luo, K. Qiu, and P. Liang, "Speeding Up Large-Scale Next Generation Sequencing Data Analysis with pBWA," *J Appl Bioinform Comput Biol 1*, vol. 1, p. 2, 2012.

[13] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[14] B. Langmead, M. C. Schatz, J. Lin, M. Pop, and S. L. Salzberg, "Searching for SNPs with cloud computing," *Genome Biol*, vol. 10, no. 11, p. R134, 2009.

[15] M. C. Schatz, "CloudBurst: highly sensitive read mapping with MapReduce," *Bioinformatics*, vol. 25, no. 11, pp. 1363–1369, 2009.

[16] Y. Li and S. Zhong, "SeqMapReduce: software and web service for accelerating sequence mapping," *Critical Assessment of Massive Data Anaysis (CAMDA) 2009*, 2009.

[17] T. Nguyen, W. Shi, and D. Ruden, "CloudAligner: A fast and full-featured MapReduce based tool for sequence mapping," *BMC research notes*, vol. 4, no. 1, p. 171, 2011.

[18] L. Pireddu, S. Leo, and G. Zanetti, "SEAL: a distributed short read mapping and duplicate removal tool," *Bioinformatics*, vol. 27, no. 15, pp. 2159–2160, 2011.

[19] R. W. Hamming, "Error detecting and error correcting codes," *Bell System technical journal*, vol. 29, no. 2, pp. 147–160, 1950.

[20] E. Ukkonen, "Approximate string-matching over suffix trees," in *Combinatorial Pattern Matching*. Springer, 1993, pp. 228–242.

[21] H. Li and R. Durbin, "Fast and accurate short read alignment with Burrows-Wheeler transform." *Bioinformatics*, vol. 25, no. 14, pp. 1754–1760, Jul 2009.

[22] A. Policriti, A. I. Tomescu, and F. Vezzi, "A randomized numerical aligner (rNA)," in *Language and Automata Theory and Applications*. Springer, 2010, pp. 512–523.

[23] F. Vezzi, C. Del Fabbro, A. I. Tomescu, and A. Policriti, "rNA: a fast and accurate short reads numerical aligner." *Bioinformatics*, vol. 28, no. 1, pp. 123–124, Jan 2012.

[24] R. M. Karp and M. O. Rabin, "Efficient randomized pattern-matching algorithms," *IBM Journal of Research and Development*, vol. 31, no. 2, pp. 249–260, 1987.

[25] T. Hoefler, C. Siebert, and W. Rehm, "A practically constant-time MPI Broadcast Algorithm for large-scale InfiniBand Clusters with Multicast," in *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*. IEEE, 2007, pp. 1–8.

[26] S. Sathe and D. Shrimankar, "Parallelization of DNA sequence alignment using OpenMP," in *Proceedings of the 2011 International Conference on Communication, Computing & Security*. ACM, 2011, pp. 200–203.

[27] L. Ligowski and W. Rudnicki, "An efficient implementation of Smith Waterman algorithm on GPU using CUDA, for massively parallel scanning of sequence databases," in *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*. IEEE, 2009, pp. 1–8.