CBIS'24
XX Congresso Brasileiro de Informática em Saúde
08/10 a 11/10 de 2024 - Belo Horizonte/MG - Brasil

A framework for counting alleles in computational clouds

Um arcabouço para contagem de alelos nas nuvens computacionais

Una herramienta para contar alelos en nubes computacionales

Marcio Nogueira Pereira Silva[1], Luís Cristóvão Pôrto[2], Alexandre da Costa Sena[3]

1 PhD student, Institute of Mathematics and Statistics, Universidade do Estado do Rio de Janeiro, Rio de Janeiro (RJ), Brasil.
2 PhD/Professor, Histocompatibility and Cryopreservation Laboratory, Universidade do Estado do Rio de Janeiro, Rio de Janeiro (RJ), Brasil.
3 PhD/Professor, Institute of Mathematics and Statistics, Universidade do Estado do Rio de Janeiro, Rio de Janeiro (RJ), Brasil.

Autor correspondente: (Prof. Dr.) Alexandre da Costa Sena
E-mail: asena@ime.uerj.br

Resumo

Objetivo: desenvolver um arcabouço para realizar comparações antropológicas, classificar alelos, inferir o tempo de espera de pacientes que necessitam de transplante, entre outras análises. Método: comparação dos alelos de doadores e pacientes através de um algoritmo de busca para, baseado na contagem dos alelos, realizar as análises. Para reduzir o tempo de execução das análises, as consultas foram otimizadas e paralelizadas, permitindo sua execução eficiente em nuvens computacionais. Resultados: o arcabouço é capaz de realizar as análises em segundos, mesmo em uma base que contém mais de 5 milhões de registros. Conclusão: a estrutura proposta tem potencial para ajudar médicos/pesquisadores a obterem informações para melhorar o processo de doação de órgãos entre doadores não aparentados.

Descritores: Alelos HLA; Nuvens Computacionais; Big Data

Abstract

Objective: develop a framework to carry out anthropological comparisons, classify alleles, infer the waiting time of patients requiring transplantation, among other analyses. Method: comparison of donor and patient alleles using a search algorithm to, based on allele counts, perform analysis. To reduce analysis execution time, queries were optimized and parallelized, allowing their efficient execution in computing clouds. Results: the framework is capable of performing analysis in seconds, even on a database that contains more than 5 million records. Conclusion: the proposed framework has the potential to help clinicians/researchers obtain information to improve the organ donation process among unrelated donors.

Keywords: HLA alleles; Computational Clouds; Big Data

Resumen

Objetivo: desarrollar un marco para realizar comparaciones antropológicas, clasificar alelos, inferir el tiempo de espera de los pacientes que requieren trasplantes, entre otros análisis. Método: comparación de alelos de donantes y pacientes mediante un algoritmo de búsqueda para realizar análisis, basado en recuentos de alelos. Para reducir el tiempo de ejecución de los análisis, se optimizaron y paralelizaron las consultas, permitiendo su ejecución eficiente en las nubes informáticas. Resultados: el framework es capaz de realizar análisis en segundos, incluso en una base de datos que contiene más de 5 millones de registros. Conclusión: el marco propuesto tiene el potencial de ayudar a los médicos/investigadores a obtener información para mejorar el proceso de donación de órganos entre donantes no emparentados.

Descriptores: Alelos HLA; Nubes Computacionales; Grandes datos

Introduction

Hematopoietic stem cell transplantation may be the only form of treatment for patients with certain types of cancer or hematological malignancies [1]. Among many factors for the success of a transplant, Human Leukocyte Antigens (HLA) matching between patient and donor is the most important [2]. While some patients find compatible

donors within their own family, many patients have to turn to voluntary donors. According to studies based on records from the United States National Bone Marrow Donor Program, 70% of patients do not have a compatible relative, depending on voluntary donors [3].

In the past, it was impossible to identify an HLA-matched donor as no system existed for recruiting, HLA typing and saving potential donor information from the population. Nowadays, there are several registries (databases) spread throughout the world whose main purpose is to encourage possible donor's recruitment and store information about their HLA typing [3]. The search for HLA donor is a complex process, that is routinely performed on donor's registries by specialized computer program in which the HLA matching algorithm can be considered the core element [4]. Through them it is possible to search for compatible donor in a deterministic fast exhaustive way [5], where the compatibility is defined by the number of alleles shared by HLA genes (e.g. alleles from *loci* HLA-A, -B, -C, -DRB1 and -DQB1) [6].

Apart from searching, most algorithms are not able to analyze an organ donor database, that is, accomplish important tasks such as: anthropological/population comparisons; comparison of the prevalence of certain alleles (allele frequency); count and classification of alleles; inferring the waiting time for patients requiring bone marrow transplantation; among other analysis.

Therefore, this work presents a framework to support doctors and researchers, through the analysis in the organ donor registry, and, therefore, help to improve the organ donation process among non-donors related. More specifically, the tool aims to compare the alleles of donors and patients, and, based on the count of these alleles, carry out anthropological/population comparisons, among other analyses. Since this analysis can be time-consuming, especially in databases with many donors (e.g. the Brazilian organ donor database contains more than 5.4 million volunteers), it is imperative that the analysis be carried out as quickly as possible. Therefore, the framework is optimized and parallel to be run on new architectures with multiple processors. Furthermore, the framework is prepared to run in any cloud to take advantage of the high availability and large amount of resources available in these environments.

Results show the feasibility of the proposed solution. The framework was able to execute several searches, where the longest took less than 40 seconds in an instance with 64 processors. Moreover, as it is prepared to be automatically executed in cloud computing environments, all searches/analysis can be carried out at the same time, creating an instance for each analysis.

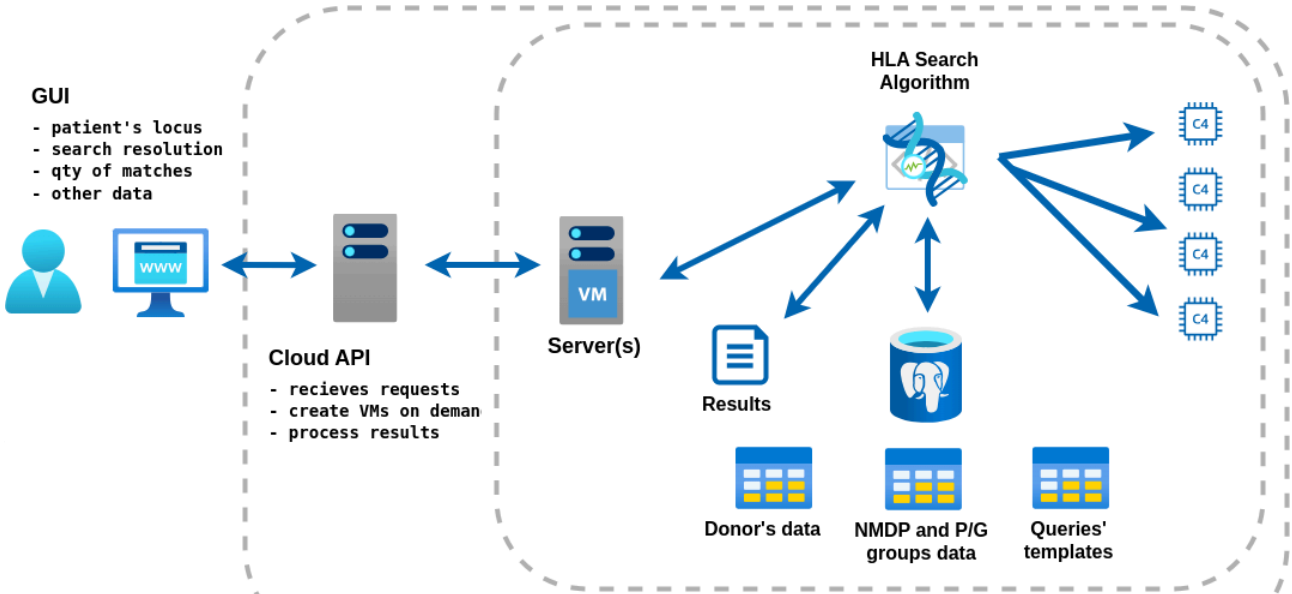
An initial prototype of the framework presented in this article can be seen in [7]. This initial version was not optimized and did not execute automatically in computing clouds. Actually, it was only a set of queries that should be manually executed in cloud or server environments. A comparison with the framework proposed in this article will be presented in the Discussion section.

On the other hand, Python for Population Genomics (PyPop) is an established software package that processes genotype and allele data and performs large-scale population genetic analyses on highly polymorphic multi-locus genotype data [8]. PyPop is not prepared to process large amount of data and, depending on the analysis, it can take a long time. Another interesting tool for advanced methods for population genetics data analysis is called Arlequin [9]. However, it can only analyze data from a population (i.e. data in a file). Actually, both PyPop and Arlequin can only handle files with at most 10.000 thousand records, which limits the scope of the analysis that can be performed.

Methods

An overview of the entire methodology adopted in this work can be seen in Figure 1 and will be detailed described in this section.

Figure 1– Overview of the methodology adopted in this work



Initially, the graphical user interface (GUI) allows users to enter all the information necessary to perform the analysis in the donor registry. Then, through the data entered into the GUI, a cloud API uses it to create the entire cloud computing infrastructure suitable for performing the query efficiently. It is important to point out that the environment is automatically created transparently to the user. Finally, the analysis is carried out in the created computational environment. The HLA search algorithm is responsible for generating the appropriate queries to compare the alleles of donors and patients, and, based on the count of these alleles generate the results. The query (i.e. task) will be executed in all processors available in the instance chosen by the cloud API, minimizing the execution time. The following subsections describe in detail each step of the methodology adopted.

Graphical User Interface

The graphical user interface allows users to enter all the information necessary to perform analysis in the donor's registry. This may be information about a single patient or a file containing multiple patients. The main input consists of HLA genotypes, that is, the alleles for each HLA *loci* being analyzed. While until recently only data from A, B and DRB1 *loci* were collected, REDOME (Brazilian Registry of Voluntary Bone Marrow Donors) currently makes available the alleles of A, B, C, DRB1, DPB1, DQA1 and DQB1 *loci*.

Another important piece of information in searches is the typing resolution. Obtaining genetic codes is subject to economic factors, as it depends on costly laboratory tests and the technique used has a great impact on the detail of the genetic code, generating records in different resolutions: low, intermediate and high. Low resolution only determines the allelic group. In turn, intermediate produces a list of possible allelic codes. Ultimately, high resolution determines the definitive allelic code. The user should also inform if the analysis will consider a full match or the number of mismatches allowed and in which *locus* it can occur. For example, considering only A, B and DRB1 *loci*, a full match will count only records where the two alleles of each *locus* are exactly the same (i.e. 6/6 match). On the other hand, a 5/6 match allows one mismatch, that is, only one of the

alleles being compared can be different. Currently, it is possible to choose the following types of matching: 5/6, 6/6, 7/8, 8/8, 9/10/,10/10, 11/12, 12/12, 13/14 and 14/14.

For more complex analyses, the user must also inform the type of query to be performed. In other words, while for simpler analysis (e.g. counting compatible donors), only the parameters mentioned in the previous paragraph are sufficient, for more complex queries (e.g. determining the classification or frequency of alleles), the type of analysis must be informed. Other information such as race/ethnicity and donor's birthplace can also be selected. These two pieces of information have a direct influence on the allele that can be found, especially in admixed populations [10].

Cloud API

The framework proposed and implemented in this work can be executed efficiently in any computer/server. However, to take advantage of the high availability and large amount of resources available in computing clouds, the framework is prepared to run in any cloud environment through the Cloud API. This API makes available to the users a web interface, enabling them to configure all necessary parameters for the analysis they wish to perform. Furthermore, this API manages the life cycle of virtual machines, including their creation, inspection, and termination, based on the user's specifications.

The Cloud API receives the search request, as can be seen in Figure 1, and, according to its parameters and the type of analysis, it creates a specific cloud instance to execute it. That is, a type of instance with the number of processors appropriate to execute the analysis as quickly as possible minimizing cost. Moreover, it prepares and initializes the entire framework so that the analysis can be carried out. Given the vast array of instance types offered by cloud providers, establishing well-defined criteria for selection is essential. To this end, comprehensive analyses were conducted using real data from previous studies, which employ various instance types. These analyses incorporated a broad spectrum of factors, including parallel processing capability, type of search required, resolution, and specific configurations of each instance. By leveraging this data, it was

created a decision-making curve that facilitates the optimal selection of instance configurations.

Therefore, to select the appropriate instance for the Cloud API, a two-step approach is implemented. Initially, as detailed in the preceding paragraph, a table was constructed listing the optimal instances for each query along with their respective parameters. In this scenario, the sole function of the cloud API is to instantiate, configure the computing environment and perform the analysis.

For queries not represented in the table, users are presented with two options: selecting a standard instance or evaluating instances to ascertain the most favorable cost-performance trade-off. The default instance is predetermined based on offering the most advantageous cost-performance ratio across a majority of analyses. Conversely, should the user opt to evaluate for the best cost-performance ratio, the Cloud API will execute a query on a subset of the input data on several predefined instances. Subsequently, it selects the instance demonstrating the optimal cost-performance ratio and updates the table to include the query, its parameters, and the identified ideal instance.

Framework for Counting HLA Alleles

After the Cloud API creates the execution environment with the ideal architecture, the proposed framework is responsible for performing the analysis, as can be seen in Figure 1 (right side). The HLA Search Algorithm has a crucial role, being responsible for generating the appropriate queries to search for the required information in accordance with the input parameters defined in the GUI. The HLA Search algorithm developed in this work has the following characteristics: (1) Deterministic: the same input must always lead to the same results; (2) Sorting: the results must be sorted according to pre-defined criteria; (3) Exhaustive: all database entries must be used in the search. (4) Scalable: database sizes can vary significantly in size and the algorithm must be able to deal with this; (5) Fast: not only is accuracy important, but also the search speed; (6) Configurable: it should be possible to define patient-donor HLA matching criteria and secondary preference criteria (gender, age, race/ethnicity, update/donor accessibility, among others).

To enhance the efficiency of the search process, the database undergoes a pre-processing phase in which auxiliary table views are generated according to specific search criteria, cataloging all existing alleles for each *locus* and processing the National Marrow Donor Program (NMDP) codes and both "P" and "G" groups. The NMDP codes constitute a classification system that assigns human leukocyte antigen (HLA) alleles for computational matching in bone marrow transplantation when the exact allele could not be determined[1]. Specifically, the "G"[2] and "P"[3] group codes categorize alleles based on their genetic and functional similarity respectively, grouping variations to enable precise matches. The systematic preprocessing of these codes and groups significantly streamlines the search for potential donors. Finally, the query is executed in parallel on the instance previously chosen by the Cloud API. It is important to highlight that the query takes advantage of all processors available, exploiting the full potential of the Virtual Machine (VM), minimizing execution time. Results are them send back to the Cloud API for further organization to be sent to the user.

Data

All data used in this study was received with an anonymized coding of records, in line with the General Law for the Protection of Personal Data (LGPD), in accordance with ethical principles outlined in the Declaration of Helsinki. Moreover, the study was approved by the Pedro Ernesto University Hospital Ethical Committee (CAAE: 56628116.3.0000.5259).

Moreover, to increase security, the file with donor allele data is always sent encrypted to the computational cloud. Furthermore, it is important to highlight that no personal data is handled by the framework. Actually, for each donor there is a numeric

[1]National Marrow Donor Program. National Marrow Donor Program — Entrusted to operate the C.W. Bill Young Cell Transplantation Program. Accessed April, 7th 2024. https://hml.nmdp.org/MacUI/

[2]HLA Nomenclature. G Codes For Reporting of Ambiguous Allele Typings. HLA Alleles. [Accessed April 7, 2024]. https://hla.alleles.org/alleles/g_groups.html

[3]HLA Nomenclature. P Codes For Reporting of Ambiguous Allele Typings. HLA Alleles. [Accessed April 7, 2024]. https://hla.alleles.org/alleles/p_groups.html

field that can only be identified by the organization responsible for the data. In other words, within the framework it is not possible to identify donors.

Donor data refers to the Brazilian National Registry of Volunteer Bone Marrow Donors (REDOME - Registro Nacional de Doadores Voluntário de Medula Óssea) database, that is the third largest bank of bone marrow donors in the world, being the largest bank with exclusively public funding [11]. At the time of this study, REDOME had 5,460,094 registered volunteers enrolled. In terms of genetic typing resolution at *locus* A, B, and DRB1, the registry includes 5.460.094 individuals classified at a low resolution, 3.913.442 at an intermediate resolution, and 63.295 at a high resolution.

Implementation of the Framework

The framework features a user-friendly, web-based graphical user interface (GUI) developed using JavaScript, HTML, and CSS. Concurrently, the cloud API is implemented using the Django web framework in Python. The cloud environment employs Amazon Elastic Compute Cloud (Amazon EC2) and the communication between the API and Amazon infrastructure is managed through the Boto3 package, an Amazon-provided library that supports Python-based integration. The database management system is PostgreSQL, installed on virtual machines equipped with the Debian operating system. Additionally, all search scripts were developed in Python and the communication between search scripts and the database is done through the psycopg package.

Results and discussion

The evaluation of the performance of the proposed and implemented framework were carried out in the AWS amazon cloud computing service. The experiments were run on AWS EC2 C7g family instances equipped with AWS Graviton 3 processors with a clock frequency of 2.6 GHz. In these instances, the amount of memory allocated is 2 GiB for each vCPU. In this work, instances of virtual machines with 1 (medium), 2 (large), 4 (xlarge), 8 (2xlarge), 16 (4xlarge), 32 (8xlarge) and 64 (16xlarge) vCPUs were used.

To analyze a wide range of options, it was executed searches with loci A, B, C,

DRB1 and DPB1, considering a full match or one mismatch allowed in one of the loci. Three groups of loci were evaluated: A, B and DRB1; A, B, C and DRB1; A, B, C, DRB1 and DPB1. Therefore, as each locus is composed of two alleles, the full match search and one mismatch search for each one of these three groups were, respectively, 6/6 and 5/6, 8/8 and 7/8, and, finally, 10/10 and 9/10. Moreover, three different resolutions were considered to observe the behavior of the framework: low, intermediate and high.

Furthermore, to investigate the scalability of the framework, searches were carried out increasing the number of patients (items) to be compared. More specifically, the framework was executed with 100, 200, 400 and 800 patients genotype to count the number of matchings in the REDOME database for each patient.

Thus, the framework was executed with all the options described previously and results can be seen in Tables 1, 2 and 3, for low, intermediate and high resolution, respectively. It is important to point out that the cloud API of the framework automatically created all the environment and is prepared to be executed in any cloud.

Figure 2 – Framework execution times (seconds) for searches in LOW resolution

Searches		medium	large	xlarge	2xlarge	4xlarge	8xlarge	16xlarge
#items	type							
100	5/6	278,74	134,73	65,66	36,66	19,49	12,91	10,28
	6/6	256,06	123,93	60,80	29,42	14,60	9,29	7,49
	7/8	109,23	53,92	27,87	13,88	7,21	4,15	2,74
	8/8	110,99	54,54	27,97	13,89	7,24	4,15	2,80
	9/10	112,77	54,89	28,18	13,96	7,38	4,17	2,82
	10/10	115,08	54,81	27,89	14,07	7,34	4,15	2,79
200	5/6	537,49	258,96	127,47	60,20	29,68	15,99	10,64
	6/6	509,10	243,66	121,74	57,43	28,26	15,21	8,47
	7/8	214,80	106,97	55,10	26,91	14,20	7,97	4,34
	8/8	219,16	107,61	55,09	27,18	14,23	7,92	4,33
	9/10	224,27	109,50	55,77	27,28	14,54	8,01	4,33
	10/10	227,28	108,77	54,78	27,18	14,50	8,00	4,38
400	5/6	1073,01	527,34	259,06	121,23	58,71	35,11	21,38
	6/6	1021,68	492,41	247,61	115,95	56,29	29,49	18,59
	7/8	430,81	214,14	110,76	53,77	28,02	15,30	8,46
	8/8	437,96	216,27	110,52	53,76	28,03	15,31	8,52
	9/10	447,89	219,77	111,80	54,06	28,40	15,41	8,51
	10/10	456,63	217,45	109,67	54,18	28,60	15,41	8,60
800	5/6	2248,32	1088,53	535,38	248,78	126,12	66,09	38,28
	6/6	2052,93	990,48	498,42	232,81	112,15	58,70	31,84
	7/8	856,59	428,74	222,44	107,20	56,12	30,38	16,83
	8/8	885,79	434,41	222,81	107,79	56,01	30,37	16,90
	9/10	890,83	441,12	223,10	108,28	56,93	30,62	17,02
	10/10	899,30	438,40	219,46	108,50	57,05	30,50	17,07

In general, the performance for the three resolutions available were very similar. Execution time decreases proportionally as the number of vCPUS available on the instance increases (lines of the tables). Conversely, the execution time increases proportionally as the number of comparisons (items) to be performed increases (columns of the tables).

When analyzing the behavior for the three groups of loci evaluated, it is clear that searches 5/6 and 6/6 are the longest, while the times for searches 7/8, 8/8, 9/10 and 10/10 are very close. More specifically, for low-resolution searches, the execution time of 5/6 and 6/6 is more than twice as long as the time for other searches. The reason for this behavior is that the number of records that have data on loci A, B and DRB1 (5/6 and 6/6 searches) is much greater than for the other loci.

Figure 3 – Framework execution times (seconds) for searches in INTERMEDIATE resolution

Searches		medium	large	xlarge	2xlarge	4xlarge	8xlarge	16xlarge
#items	type							
100	5/6	207,25	100,53	48,74	23,20	11,60	6,61	4,06
	6/6	203,36	99,39	49,35	23,10	11,53	6,62	4,10
	7/8	105,03	51,82	26,51	13,25	7,02	4,09	2,71
	8/8	107,05	52,77	26,76	13,41	6,96	4,13	2,75
	9/10	107,30	52,87	26,30	13,51	7,13	4,04	2,73
	10/10	107,77	53,22	26,41	13,42	7,18	4,13	2,76
200	5/6	411,85	199,67	96,84	45,50	22,56	12,47	6,84
	6/6	404,87	197,21	97,41	45,98	22,48	12,46	6,87
	7/8	211,32	104,18	52,95	26,56	13,93	7,72	4,35
	8/8	215,92	105,93	53,51	26,63	13,93	7,68	4,39
	9/10	215,77	106,55	52,45	26,81	14,20	7,76	4,42
	10/10	215,36	106,94	52,69	26,98	14,26	7,77	4,44
400	5/6	823,72	399,39	194,67	90,42	44,57	23,88	13,19
	6/6	812,37	395,30	196,02	91,16	44,76	23,85	13,20
	7/8	419,37	208,53	106,03	52,27	26,92	14,84	8,28
	8/8	430,25	212,24	107,06	52,64	27,07	14,88	8,34
	9/10	430,74	212,35	105,12	52,73	27,48	14,97	8,32
	10/10	428,06	214,16	105,22	53,11	27,58	15,01	8,31
800	5/6	1636,57	805,46	394,40	182,67	89,40	47,04	27,38
	6/6	1637,85	791,86	396,05	183,13	89,31	47,20	27,49
	7/8	842,11	421,34	214,09	105,03	55,64	29,54	19,49
	8/8	862,36	428,93	216,15	106,01	55,84	29,53	19,65
	9/10	857,29	427,93	213,14	106,24	56,53	29,81	19,64
	10/10	867,92	431,18	212,51	106,69	56,73	30,05	19,69

Similarly, when comparing the executions times in low, intermediate and high resolutions, it is clear that searches in low resolution are the longest, while intermediate searches are a little faster and, finally, high resolution searches are the fastest (e.g. up to 10 times faster than 5/6 search with 800 items in low resolution). The reason for that behavior is also the number of records in each of resolutions, as described in the Data section. Analyzing the scalability of searches, it is possible to see that for low and intermediate resolutions the behavior was very consistent, decreasing linearly according to the increase in the number of processors. On the other hand, for high-resolution searches, with the exception of searches 5/6 and 6/6, the use of 32 and 64 processors did not improve performance or improved it very little.

Figure 4 – Framework execution times (seconds) for searches in HIGH resolution

Searches		medium	large	xlarge	2xlarge	4xlarge	8xlarge	16xlarge
#items	type							
100	5/6	32,79	14,31	8,35	8,84	5,87	4,04	2,81
	6/6	35,61	13,89	7,82	9,12	4,20	4,04	3,46
	7/8	9,49	5,04	3,84	1,47	0,86	0,74	0,62
	8/8	15,80	6,84	4,78	2,32	2,00	2,14	3,49
	9/10	9,72	5,13	3,90	1,50	0,88	0,78	0,60
	10/10	16,05	6,96	4,83	2,33	2,01	2,16	3,48
200	5/6	65,30	34,02	17,79	13,54	4,07	5,32	3,30
	6/6	72,60	38,10	18,93	14,31	8,87	5,40	3,59
	7/8	21,14	10,58	9,20	2,89	1,63	1,28	1,11
	8/8	34,63	16,01	10,77	5,59	7,85	3,51	3,58
	9/10	21,62	10,80	9,34	2,93	1,67	1,30	1,20
	10/10	35,12	16,24	10,90	5,63	7,86	3,51	3,61
400	5/6	118,16	65,54	32,54	26,42	7,82	5,57	4,09
	6/6	142,72	74,40	35,07	28,05	9,53	6,32	4,29
	7/8	44,94	22,95	16,98	6,17	3,24	3,07	3,02
	8/8	76,49	34,48	20,98	8,62	7,81	4,31	4,28
	9/10	45,92	23,41	17,24	6,28	3,25	3,12	3,02
	10/10	77,58	34,95	21,18	8,74	7,85	4,32	4,26
800	5/6	211,07	128,91	64,11	49,99	13,32	9,13	4,80
	6/6	252,09	147,38	68,62	52,35	19,62	13,15	8,55
	7/8	85,71	43,55	32,73	12,15	6,01	4,23	4,00
	8/8	134,02	66,65	39,61	16,65	15,27	10,23	8,31
	9/10	87,60	44,47	33,27	12,40	6,06	4,25	4,01
	10/10	136,10	67,60	40,16	16,92	15,40	10,33	8,29

Discussion

This work implemented an optimized framework for compare the alleles of donors and patients, and, based on the count of these alleles, carry out analyses. Due to the large number of records (i.e. more than 5.400.000), the framework was optimized to not only speed up queries, but to perform them in parallel. Furthermore, it is prepared to take advantage of the high availability and large amount of resources available in computing clouds, automatically executing in these environments.

It was not possible to compare the results of this work with PyPop [7] and Arlequim [8], since these tools cannot process more than 10.000 records and the REDOME database has more than 5,4 million donors. In turn, when comparing the results of the framework with an initial version (prototype) [11], it is clear the improvement of the new version. While the prototype version took 1012 seconds to perform a 5/6 search in low resolution and 7424 seconds for a 5/6 search in intermediate resolution considering 100 patients (items) in a medium instance (1 vCPU), the framework proposed in this work took only 278 and 207 seconds to execute the same searches, respectively. That is, this new version was 3.7 and 33.9 times faster than the prototype running in 1 vCPU.

It is important to point out that the high availability of computing clouds allows that all searches can be executed at the same time. Therefore, for example, all searches in low resolution for 800 patients (i.e. 5/6, 6/6, 7/8, 8/8, 9/10 and 10/10) could be executed together, each one in a different virtual machine, and the execution time to achieve that would be only 38 seconds (i.e. the execution time of the slowest query).

The cloud API was able to choose the right/best instance for each search. While for low and intermediate resolution instances with the largest number of processors were selected (16xlarge), for high resolution analysis only for queries 5/6 and 6/6 the 16xlarge instance was chosen. For the other queries, 4xlarge or 8xlarge instances were selected based on performance on each of these instances.

Finally, it is important to point out that, currently, the framework only contain donor data. However, any researcher using the framework can use patient data to compare with donor data. For example, Nunes' work [10] used the proposed framework and compared the alleles of patients with Sickle Cell Disease with the REDOME donor's alleles and identified

that patients with a higher African ancestry face greater difficulty in locating potential donors.

Conclusion

This work implemented and evaluated an optimized parallel framework for comparing HLA alleles and carry out analyses. Unlike existing tools that cannot handle more than 10.000 records, the proposed framework is not only capable of working with millions of records, but also performing queries very quickly (e.g. The longest query took just 38 seconds on a machine with 64 processors). Therefore, this work can support doctors and researchers, through the analysis of patterns in the organ donor registry, and, therefore, help to improve the organ donation process among non-donors related.

Acknowledgements

This research was supported by the projects funded by CNPq/AWS (Process Number 421828/2022-6) and FAPERJ APQ1 26/2021 (Process Number E-26/211.798/2021).

References

1. Singh AK, McGuirk JP. Allogeneic Stem Cell Transplantation: A Historical and Scientific Overview. Cancer Research (2016) 76:6445-6451.

2. Tiercy JM. How to select the best available related or unrelated donor of hematopoietic stem cells? Haematologica (2016) 101(6):680-687.

3. Gragert L. et al. HLA match likelihoods for hematopoietic stem-cell grafts in the U.S. registry. New England Journal of Medicine, 2014, 371(4):339-348.

4. Burns LJ, Miller JP, Confer DL. Past, present and future of the national marrow donor program. Revista de Hematología. 2016;17(3):195-204.

5. Dehn J. et al. HapLogic: A predictive human leukocyte antigen matching algorithm to enhance rapid identification of optimal unrelated hematopoietic stem cell sources for transplantation. Biology of Blood and Marrow Transplantation. 2016, 22(11):2038-2046.

6. Bochtler, W. et al. A Comparative Reference Study for the Validation of HLA-Matching Algorithms in the Search for Allogeneic Hematopoietic Stem Cell donors and cord blood units. HLA. 2016, 87(6).

7. Marcio N. P. Silva, Karla Figueiredo, Luís C. M. S. Pôrto, Alexandre C. Sena. Uma Abordagem para Análise de Padrões em Banco de Dados de Doadores de Órgãos. Anais do Simpósio Brasileiro de Computação Aplicado a Saúde. 2022.

8. Lancaster AK, Single RM, et al. PyPop: A mature open-source software pipeline for population genomics. *Frontiers in Immunology*. 2024, 15.

9. Excoffier L, Lischer HE. Arlequin suite ver 3.5: a new series of programs to perform population genetics analyses under Linux and Windows. Mol Ecol Resour. 2010, 10(3):564-567.

10. Nunes K, Aguiar V, Silva M. et al. How Ancestry Influences the Chances of Finding Unrelated Donors: An Investigation in Admixed Brazilians. Frontiers in Immunology. 2020, 11.

11. REDOME - Registro Nacional de Doadores Voluntários de Medula Óssea. 2024.