Big Data Analytics Programming: Assignment 2 Report

Fotios Kalioras

1 SIMD

Following the assignment instructions, both scalar and SIMD versions of the 1NN algorithm were implemented. The SIMD version utilizes the AVX registers until fewer than 8 columns remain in the dataset, at which point it switches to scalar mode. Due to the fact that the number of columns is not always a multiple of 8, aligned access cannot always be guaranteed, resulting in unaligned accesses that introduce a slight performance penalty. Additionally, the row-major approach was chosen to take advantage of row-level caching. Lastly, since Euclidean distance comparison only needs relative distances, square rooting is unnecessary [1]. Therefore squared distances are compared directly in order to skip useless computing.

1.1 Experiment Setup

For this experiment, the main function of find_closest.cpp was modified and a driver.py script was developed to perform 100 measurement repetitions for each CSV file. The decision to handle repetitions outside the C++ program was made to prevent cache warm-ups, which could bias subsequent function calls. By invoking the program from the Python script, we ensure consistent conditions throughout the experiment. Additionally, the program's output was adjusted to a format that is easy to parse by the Python script. Since using the departmental computers is not mandatory in this task, all experiments were conducted on a home PC with an Intel i7-8700H processor running macOS.

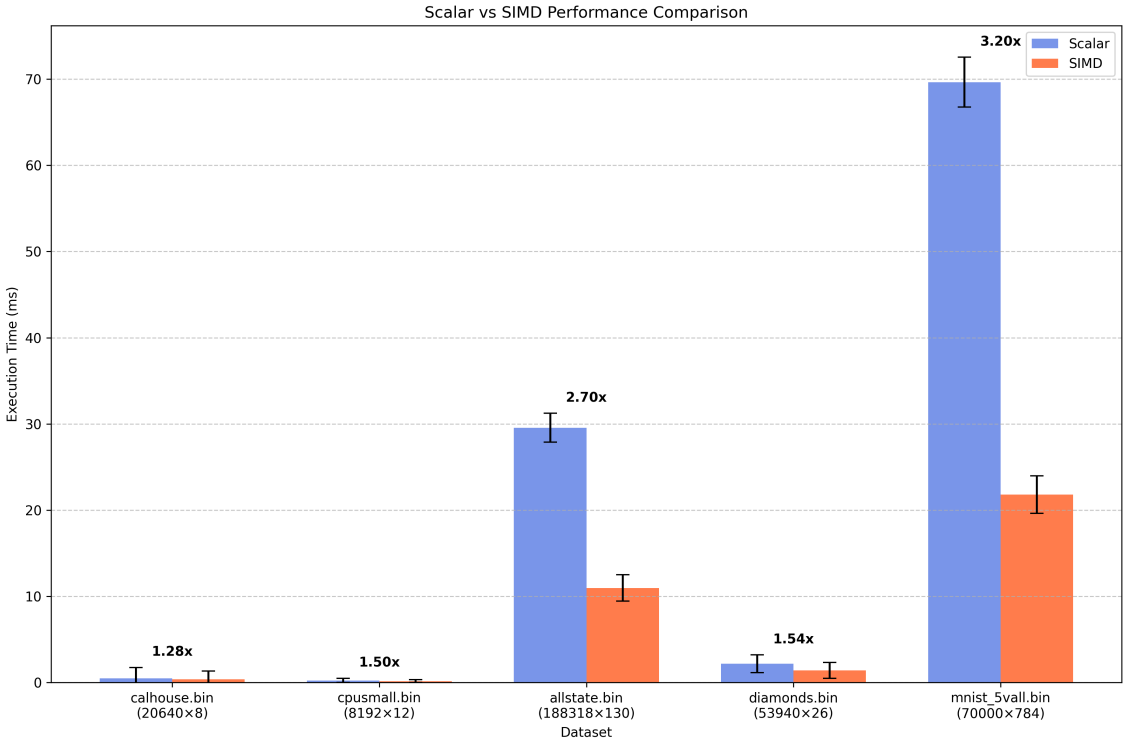


Figure 1: Execution times and speedups for different datasets using SIMD and scalar algorithms

1.2 Experiment results and discussion

Figure 1 presents the execution times for each dataset and algorithm combination. The results align with expectations; datasets with a larger number of columns show the most significant performance improvements. For example, the MNIST dataset with 784 columns achieves a 3.2x speedup, while smaller datasets experience

less pronounced gains. Interestingly, even the Callhouse dataset, which has only 8 columns with 32-bit single-precision floating point numbers (the width of an AVX register [2]), benefits from SIMD despite the inherent overhead of AVX operations.

2 Spark

Following the assignment the requested Spark programs were created. Each process is documented in detail and extra care was taken to prevent unnecessary functions that would hurt the performance.

2.1 Experiment Setup

Since timing is not the primary focus for this part of the assignment, but rather the accuracy of the results, an initial rigorous analysis was conducted on the sample game dataset to identify and account for all edge cases. To gain a deeper understanding of the data, the tool from the NBA-Player-Movements GitHub repository by linouk23 [3] was used. This tool not only visualizes each individual event but also extracts the corresponding basket coordinates, which are (5.5, 25) and (89, 25) (in inches).
Figure 2 shows a screenshot of the tool, which plots the positions of the players and the ball, along with the players' names. The coordinates displayed in the bottom right corner correspond to the left basket.

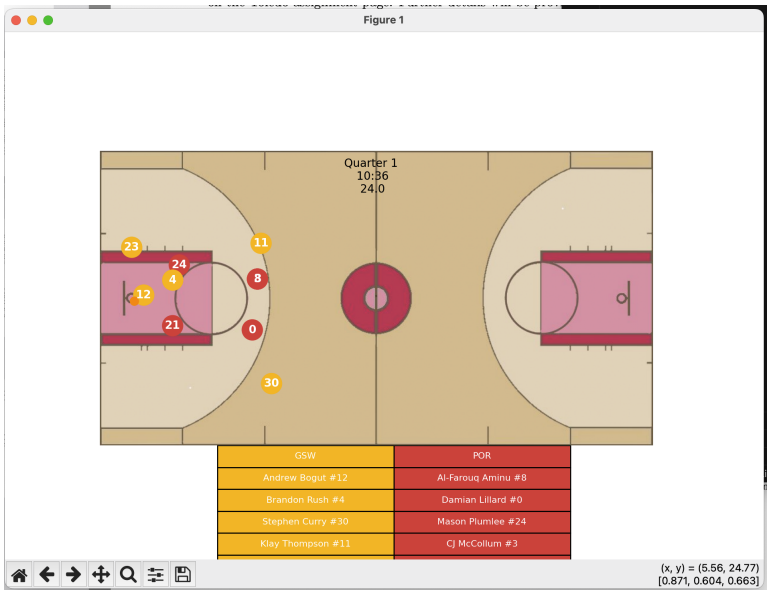


Figure 2: Screenshot of the NBA tool with the coordinates of the left basket

2.2 Some comments about dataset ordering

Upon reviewing the dataset and experimenting with an initial version of the code, two key issues were identified:

- The moments dataset contains multiple similar entries, each involving the same player at the same position and timestamp, but corresponding to different events. This suggests that events are not isolated but rather overlap.
- The game clock during a quarter does not behave like a typical timer, exhibiting "jumps" of up to 10 seconds. Additionally, there are instances where the clock is stopped, making it unreliable for deterministic ordering. As a result, entries with the same game_clock can be scattered across the cluster, losing their temporal continuity.

From these findings two manipulations were made:

- For the first two questions, duplicate rows were dropped without concern for which specific entries were removed, as the focus was solely on the player positions. For the final question, the relevant rebound events were manually selected.
- Since neither game_clock nor shot_clock could be used to reliably order the dataset, a decision was made to append a monotonically increasing ID to each entry. While this introduces a slight performance penalty, it ensures that the natural order is maintained, even when the dataset is sharded.

2.3 Speed Zones

2.3.1 A comment about the clock approach

Using a window that splits the data by player, game and quarter, we cover all potential cases for the end of a speed zone. The only remaining rare case is when a player exits and later returns within the same quarter. To account for this, a flagging approach was implemented to detect a jump of more than 15 seconds on the clock. The assumption here is that a player cannot be substituted for less than 15 seconds. This approach prevents false positives from clock jumps, as discussed earlier, while also capturing these rare substitution events. It is important to note that for players substituted during a quarter but not returning, their run is considered ended, as there will be no further data available to compute the moving average.

2.3.2 A comment about speed zones

During the analysis of the results, which will be reported below, it was noted that many players did not reach the "fast" speed zone, prompting further investigation. Upon review, it was found that one of the quickest movements in basketball is the fast break [4]. According to [5], the fastest players have an average fast break speed of 8.8 m/s. Therefore, setting the"fast" filter to 8 m/s only captures the fastest players, excluding others who may still exhibit significant speed during gameplay. it should be further noted that if the filter is reduced to 7 m/s many more players reach the threshold.

2.3.3 Speedzones reporting

The analysis shows that many players had the median number of highest speed runs, 3. As a tiebraker, the player with the most distance covered in these runs was selected. The player is Kristaps Porzingis (ID: 204001) with a total covered distance of 28.84m in the fast zone. Additionally, the player with the number of most speed runs, 71, is Draymond Green (ID: 203110), who, according to a quick Google search, is known for being a very fast power forward. Many people praise his speed, although some criticize his fast regression due to his age. Therefore, this result seems reasonable. Additionally, Green covers the greatest distance in the high-speed zone, at 108.32 meters. Lastly, 95 players never entered the "fast" speed zone.

2.4 Rebounds

2.4.1 A comment about rebounds

During the analysis of the results, it was observed that some players recorded rebound distances of up to 13 meters. Given that the maximum three-point distance from the basket is approximately 7.3 meters, as shown in Figure 3, this warranted further investigation.

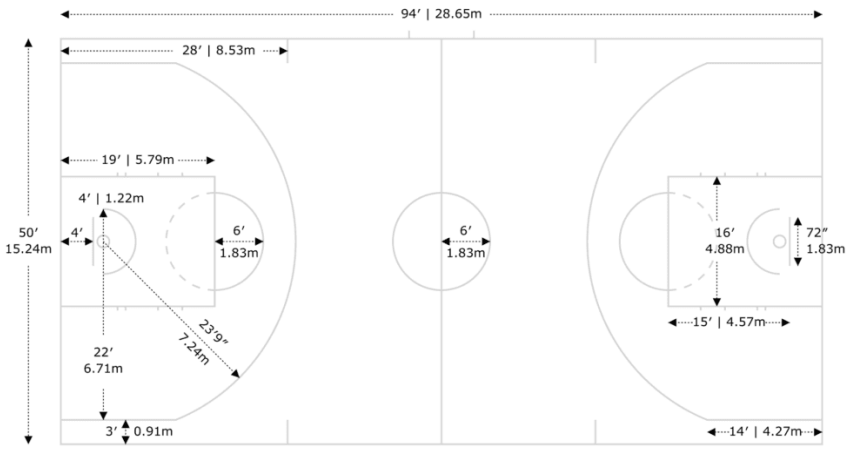


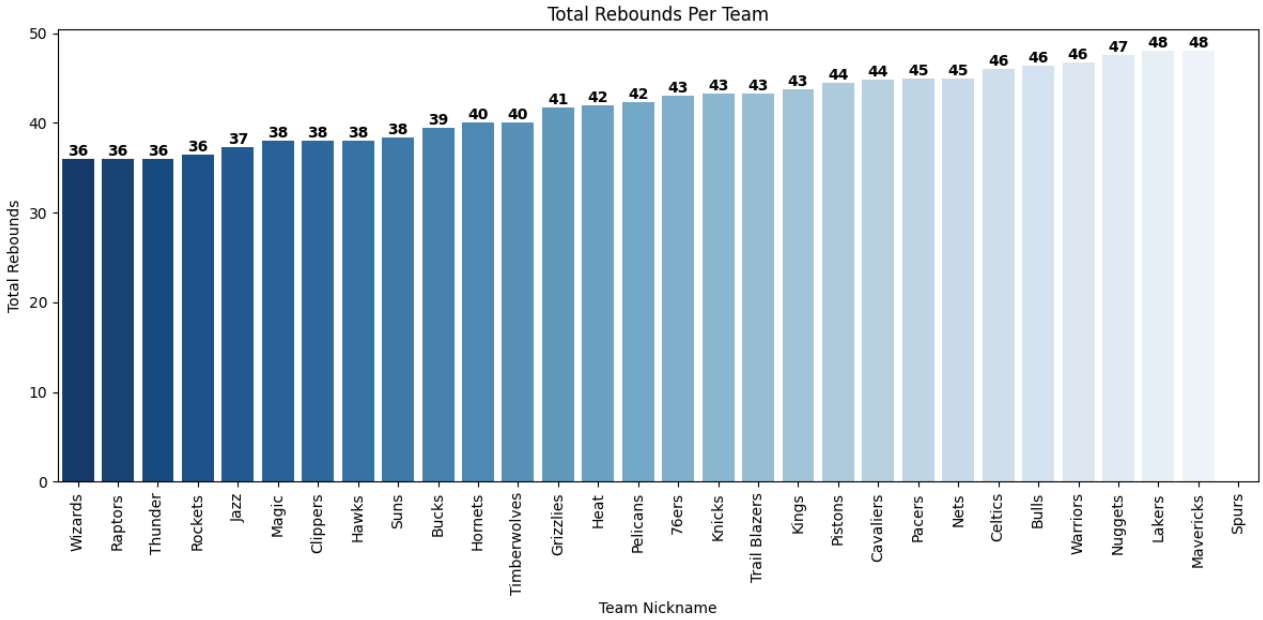
Figure 3: Useful court dimensions that were used during the development process

It was found that certain team rebounds, resulting from contested plays where the ball goes out of bounds, were incorrectly attributed to individual players. Consequently, when these players reentered the court, typically from the middle, the system registered their return as a rebound. While this contradicts NBA rebounding rules [6], SportVU still classifies these instances as rebounds. Therefore, it was decided to retain these recorded rebounds without modification.

2.4.2 A comment about Spurs

Upon plotting the average rebounds per game for each team, it was observed that the Spurs did not have an entry. This led to further investigation, including retrieving a complete list of NBA games played between October 27, 2015 and January 23, 2016, the period covered by our dataset. The analysis revealed that during this timeframe, the only game involving the Spurs against either the Warriors or the Cavaliers was a matchup between the Spurs and Cavaliers on January 14, 2016. However, according to a footnote in the assignment, this game is missing from the dataset. Consequently, the absence of data for the Spurs is expected.

2.4.3 Average number of rebounds per match for each team



2.5 Players and distances of further rebounds

	player_ID	distance
1	201939	13.407467162359744
2	204025	12.855401170478284
3	201935	12.785161270138364
4	200751	12.627128843716694
5	2747	12.6191501325933
6	201937	12.526052257647038
7	101161	12.4323079264162
8	2592	12.374876138440811
9	201177	12.331155834680786
10	2544	11.996593188909753

References

[1] Oracle Corporation. *Euclidean and Euclidean Squared Distances*. Accessed: 2025-03-20. 2025. URL: `https://docs.oracle.com/en/database/oracle/oracle-database/23/vecse/euclidean-and-squared-euclidean-distances.html`.

[2] Wikipedia contributors. *Advanced Vector Extensions — Wikipedia, The Free Encyclopedia*. [Online; accessed 14-March-2025]. 2025. URL: `https://en.wikipedia.org/w/index.php?title=Advanced_Vector_Extensions&oldid=1277724286`.

[3] Kostya Linou. *NBA-Player-Movements*. `https://github.com/linouk23/NBA-Player-Movements`. 2016.

[4] Basketball Universe. *Fast Break Basketball: Definition, History, and Strategies*. 2025. URL: https://basketballuniverse.io/fast-break-basketball/.

[5] Reddit User. *What speed do NBA players run at during a fast break? In mph*. Online forum post. June 2022. URL: https://www.reddit.com/r/nba/comments/vky9ze/what_speed_do_nba_players_run_at_during_a_fast/.

[6] NBA Video Rulebook. *Team Rebound, No Individual Rebound as Ball Goes Out of Bounds*. https://videorulebook.nba.com/archive/team-rebound-no-individual-rebound-as-ball-goes-out-of-bounds/. Accessed: 2025-03-13.