K-Nearest Neighbors

CSE 633

Abel Jacob

Agenda

- Working of KNN
- Sequential Algorithm
- Need for parallelization
- Parallel Algorithm
- Dataset
- Testing Results

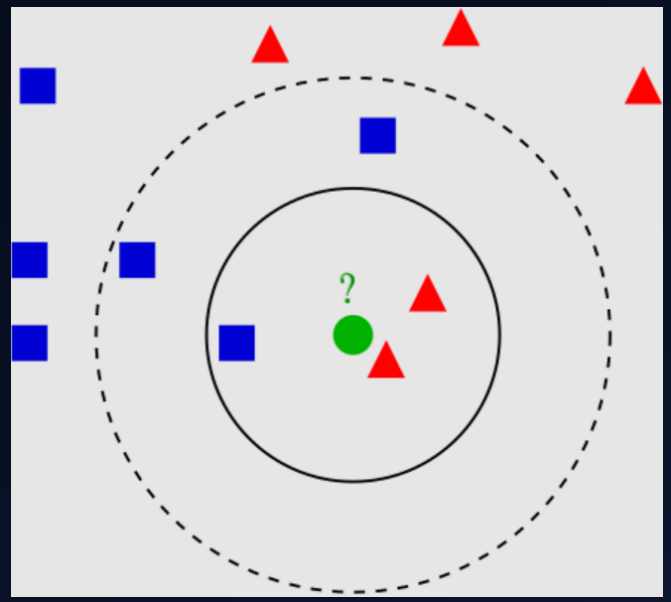
Source: ibm.com (reference 1)

Working of KNN

- Supervised learning
- Classification or regression
- Selection of K (K=5 for testing)
- Assign class labels
  - Based on distance measure
  - Majority vote
- Training/Testing

$$d(x,y) = \sqrt{\sum_{i=1}^{n} (y_i - x_i)^2}$$

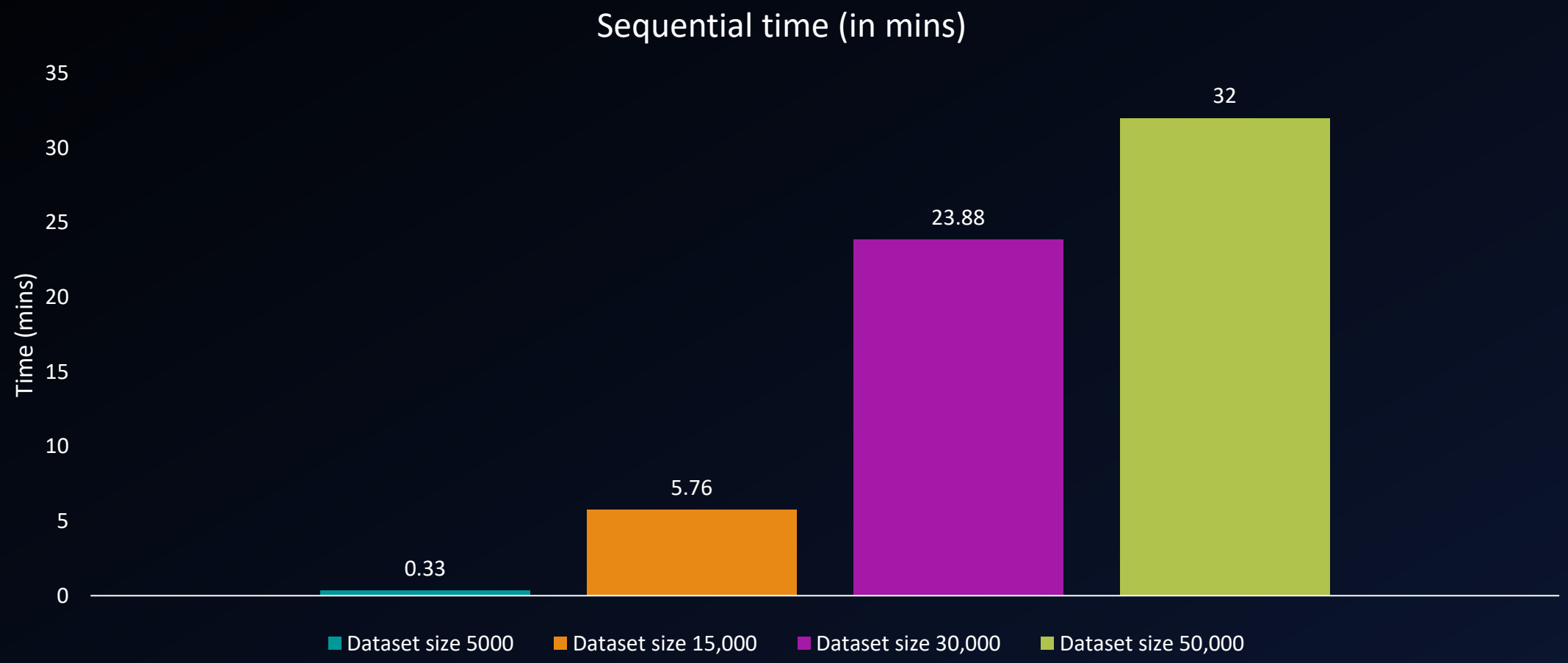
Source: ibm.com
(reference 1)



Source: wikipedia.org
(reference 2)

Sequential Algorithm

- Training:
  - Fitting the training data and its class labels
- Choose K
- Check distance of test data points with all training data
  - Check k nearest neighbors to that point
  - Predict class label
- Accuracy
  - Compare predicted class labels with actual labels



Prediction Accuracy: ~91 %

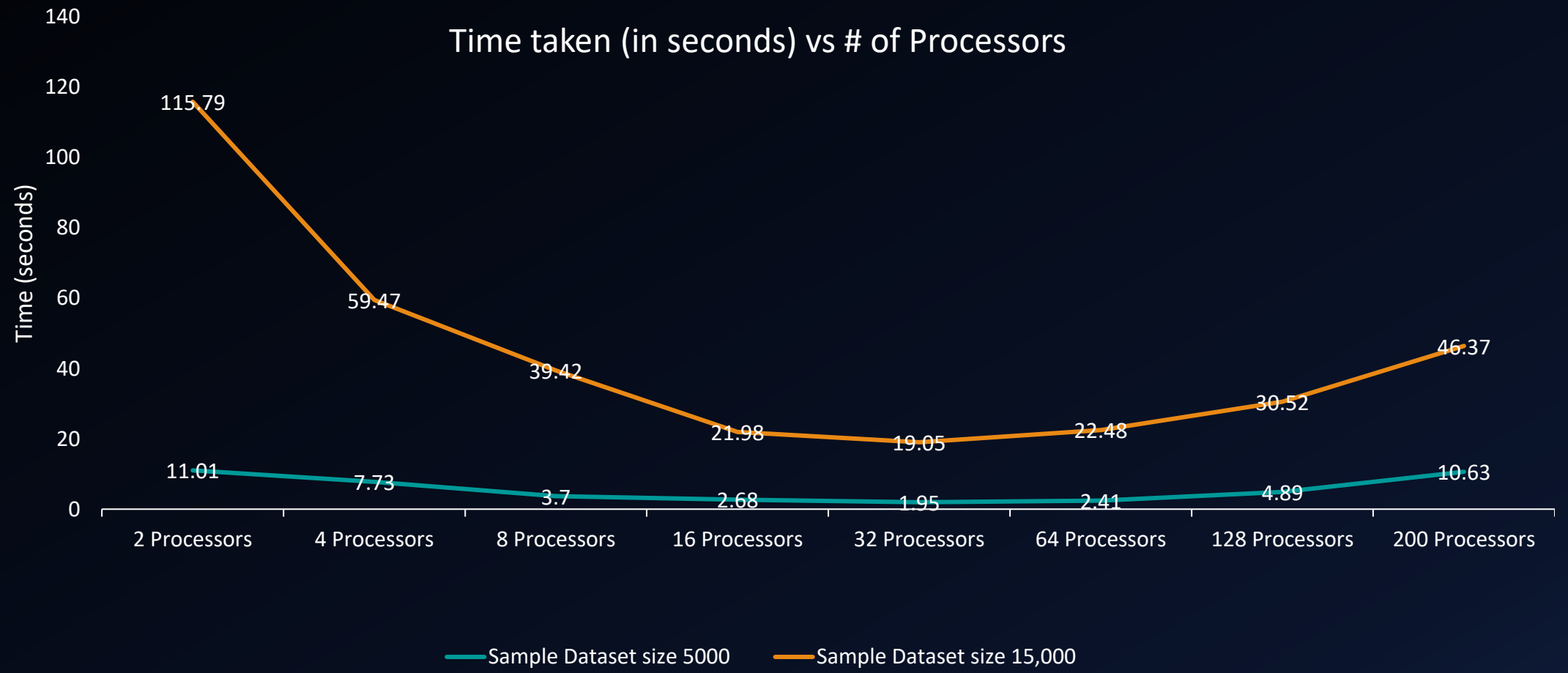
Need for parallelization

- Performance of sequential degrades as dataset size increases
- Sequential algorithm is known to be computationally and data intensive
- Parallelize to improve efficiency

Parallel Algorithm

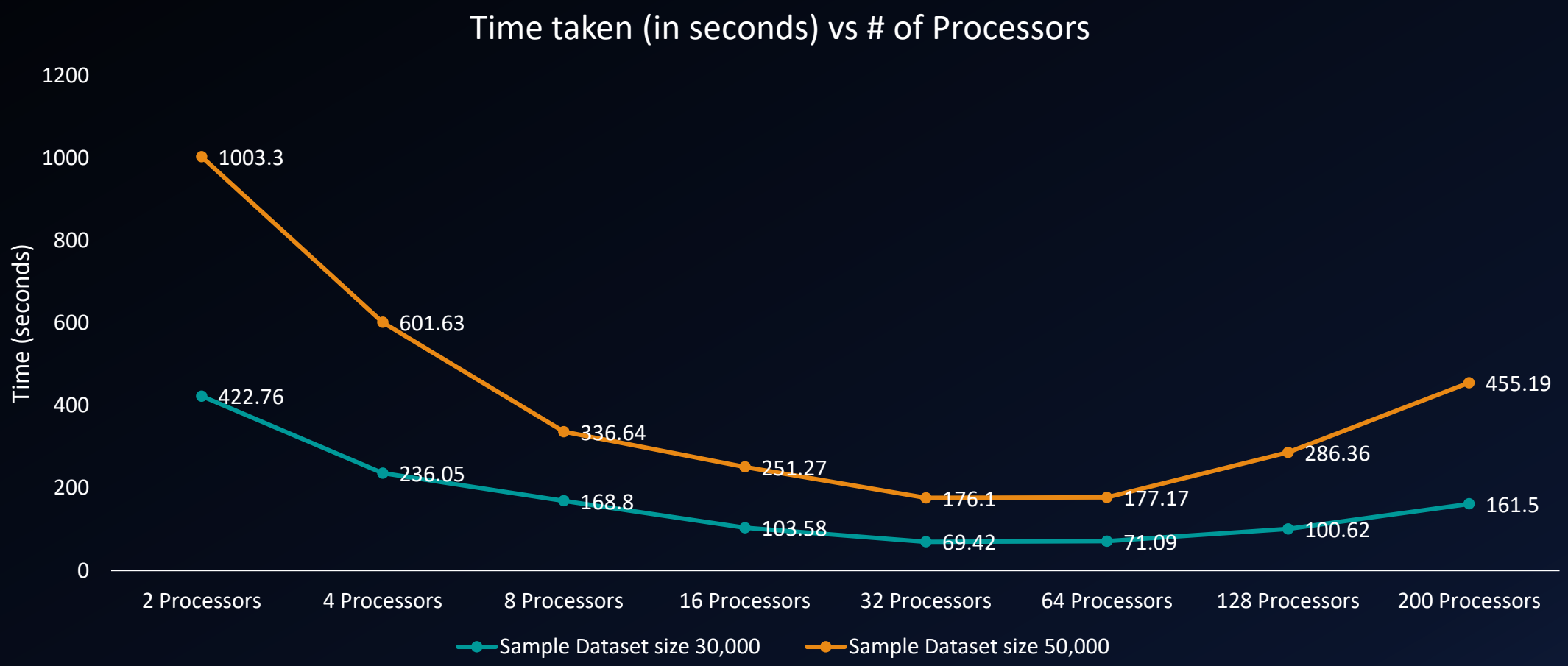
- '**Scatter**' training data to the processes
- '**Broadcast**' test data point to each process
- Calculate Euclidian distance for this point on set of processes
- '**Gather**' all the distances and sort
- Predict the class labels for this data point
- Repeat for remaining test data
- Check accuracy of predicted labels with test data labels

Multiple CPUs per node



Prediction Accuracy: ~91 %

Multiple CPUs per node



Prediction Accuracy: ~91 %

Sbatch script

- #SBATCH --**exclusive**
- #SBATCH --constraint=**CPU-E5-2650v2&IB**
- #SBATCH --nodes=16
- #SBATCH --ntasks-per-node=1

Soccer dataset

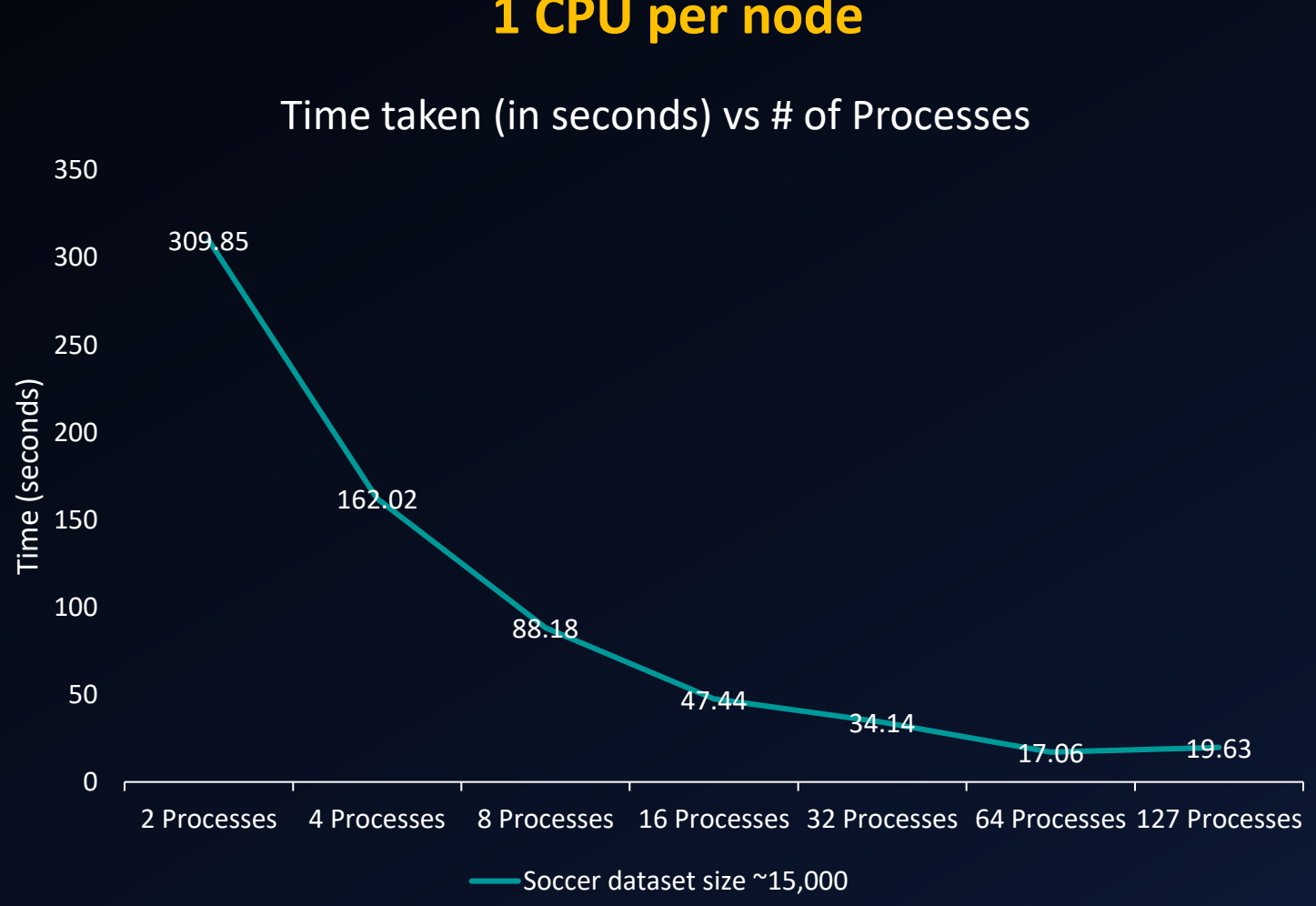
- Soccer dataset with ~25k matches
- Used cleaned/processed version with ~15k rows on sequential and parallel KNN model
- Predict match outcome (Win/Loss/Draw)
- Accuracy around ~45%

	8135	15639	2958
home_team_goals_difference	12.0	-6.0	-6.0
away_team_goals_difference	4.0	19.0	-5.0
games_won_home_team	6.0	2.0	2.0
games_won_away_team	5.0	7.0	2.0
games_against_won	0.0	0.0	1.0
games_against_lost	2.0	2.0	1.0
League_1.0	False	False	False
League_1729.0	False	False	True
League_4769.0	False	False	False
League_7809.0	True	False	False
League_10257.0	False	False	False
League_13274.0	False	False	False
League_15722.0	False	False	False
League_17642.0	False	True	False
League_19694.0	False	False	False
League_21518.0	False	False	False
League_24558.0	False	False	False
home_player_1_overall_rating	77.0	70.0	79.0
home_player_2_overall_rating	67.0	69.0	68.0
home_player_3_overall_rating	76.0	68.0	77.0
home_player_4_overall_rating	76.0	69.0	77.0
home_player_5_overall_rating	73.0	65.0	74.0
home_player_6_overall_rating	77.0	64.0	73.0

Execution times for soccer dataset

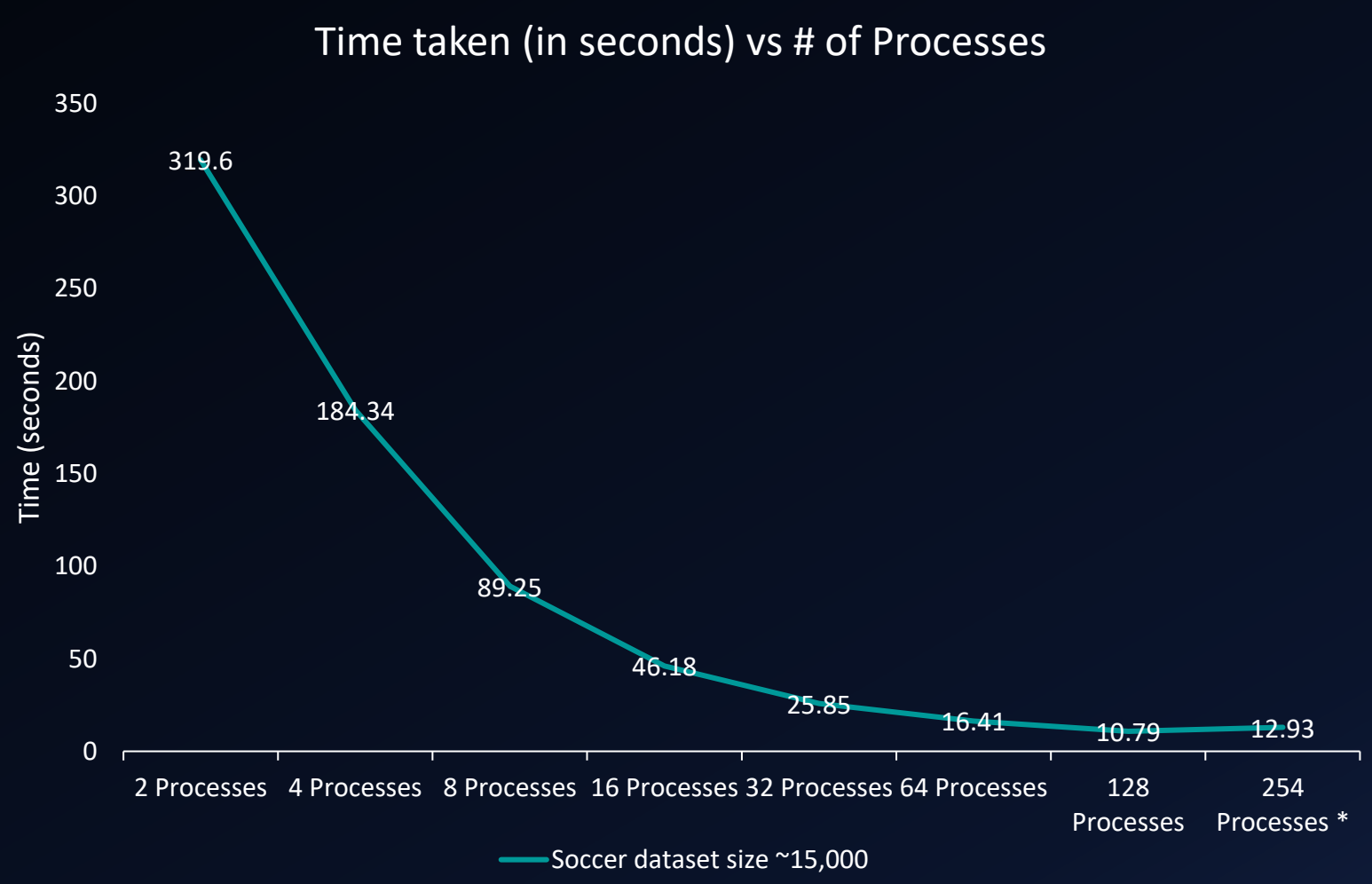
CPUs per node * number of nodes	Total number of Processes	Time (in seconds)
1 * 2	2	309.85
1 * 4	4	162.02
1 * 8	8	88.18
1 * 16	16	47.44
1 * 32	32	34.14
1 * 64	64	17.06
1 * 127	127 *	19.63

* as 128 nodes NA for CPU-E5-2650v2&IB, 127 used instead



CPUs per node * number of nodes	Total number of Processes	Time (in seconds)
2 * 2	2	319.6
2 * 2	4	184.34
2 * 4	8	89.25
2 * 8	16	46.18
2 * 16	32	25.85
2 * 32	64	16.41
2 * 64	128	10.79
2 * 127	254 *	12.93

* as 128 nodes NA for CPU-E5-2650v2&IB, 127 used instead

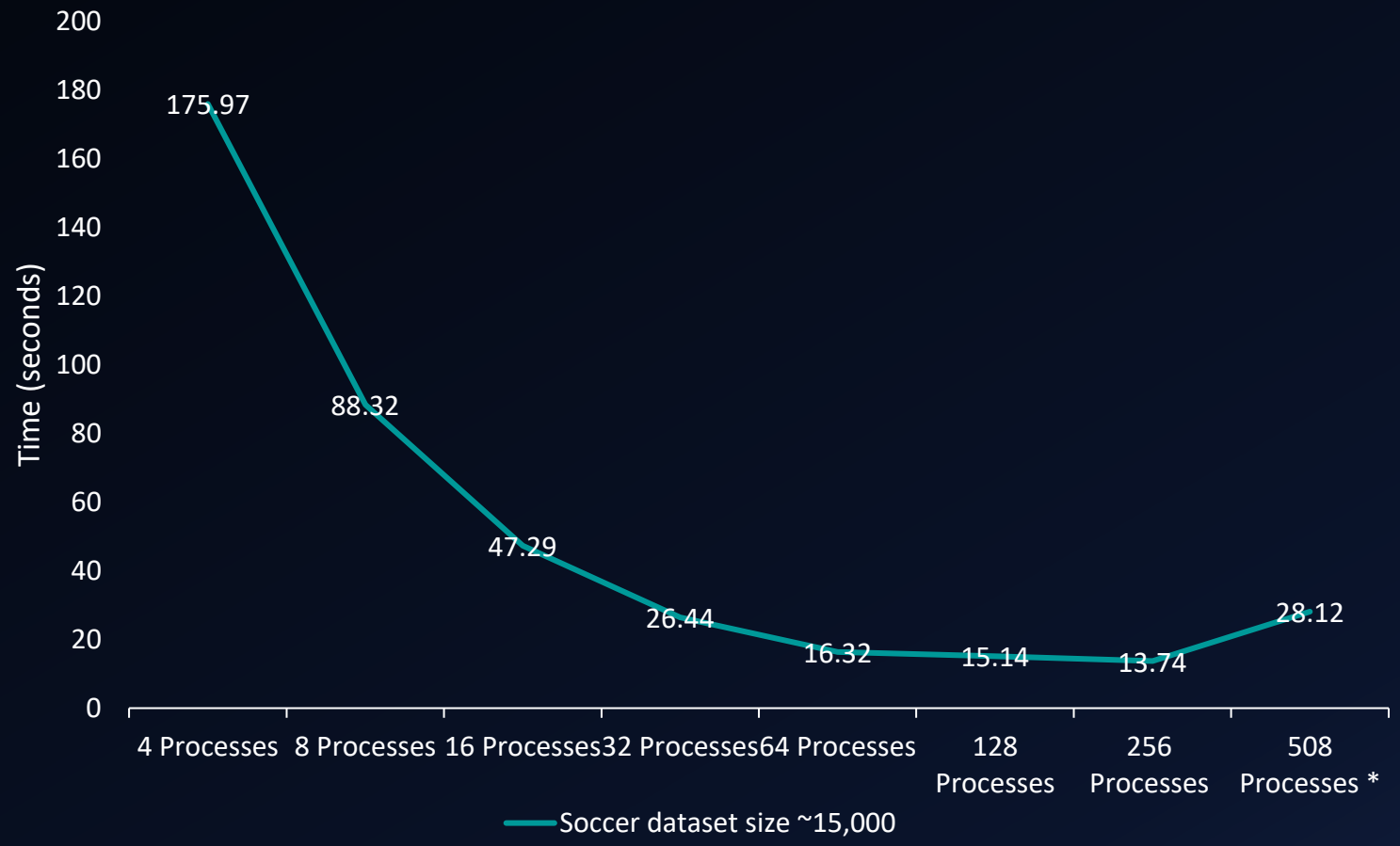


CPUs per node * number of nodes	Total number of Processes	Time (in seconds)
4 * 1	4	175.97
4 * 2	8	88.32
4 * 4	16	47.29
4 * 8	32	26.44
4 * 16	64	16.32
4 * 32	128	15.14
4 * 64	256	13.74
4 * 127	508 *	28.12

* as 128 nodes NA for CPU-E5-2650v2&IB, 127 used instead

4 CPUs per node

Time taken (in seconds) vs # of Processes



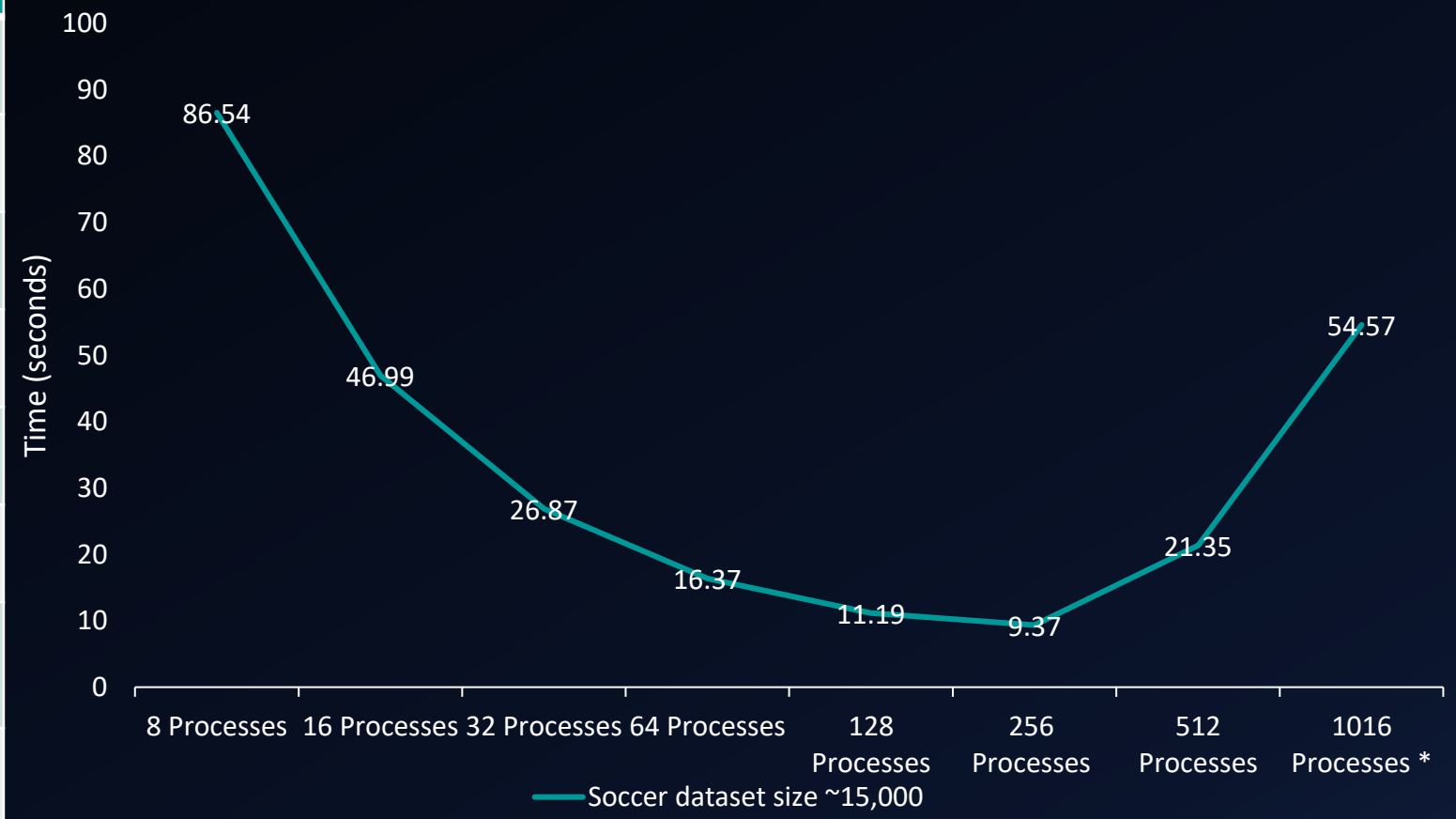
Prediction Accuracy: ~45 %

CPUs per node * number of nodes	Total number of Processes	Time (in seconds)
8 * 1	8	86.54
8 * 2	16	46.99
8 * 4	32	26.87
8 * 8	64	16.37
8 * 16	128	11.19
8 * 32	256	9.37
8 * 64	512	21.35
8 * 127	1016 *	54.57

* as 128 nodes NA for CPU-E5-2650v2&IB, 127 used instead

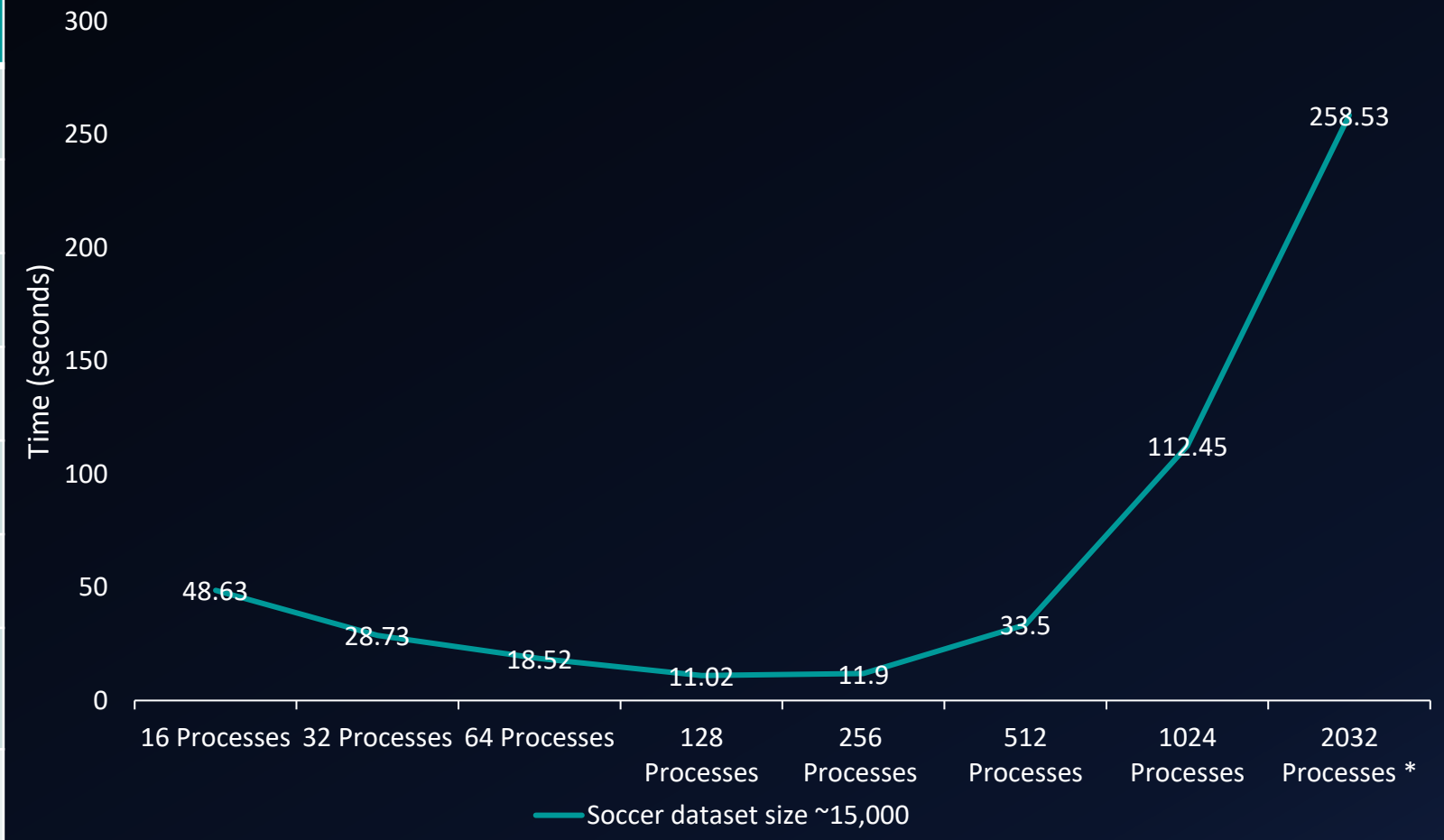
8 CPUs per node

Time taken (in seconds) vs # of Processes



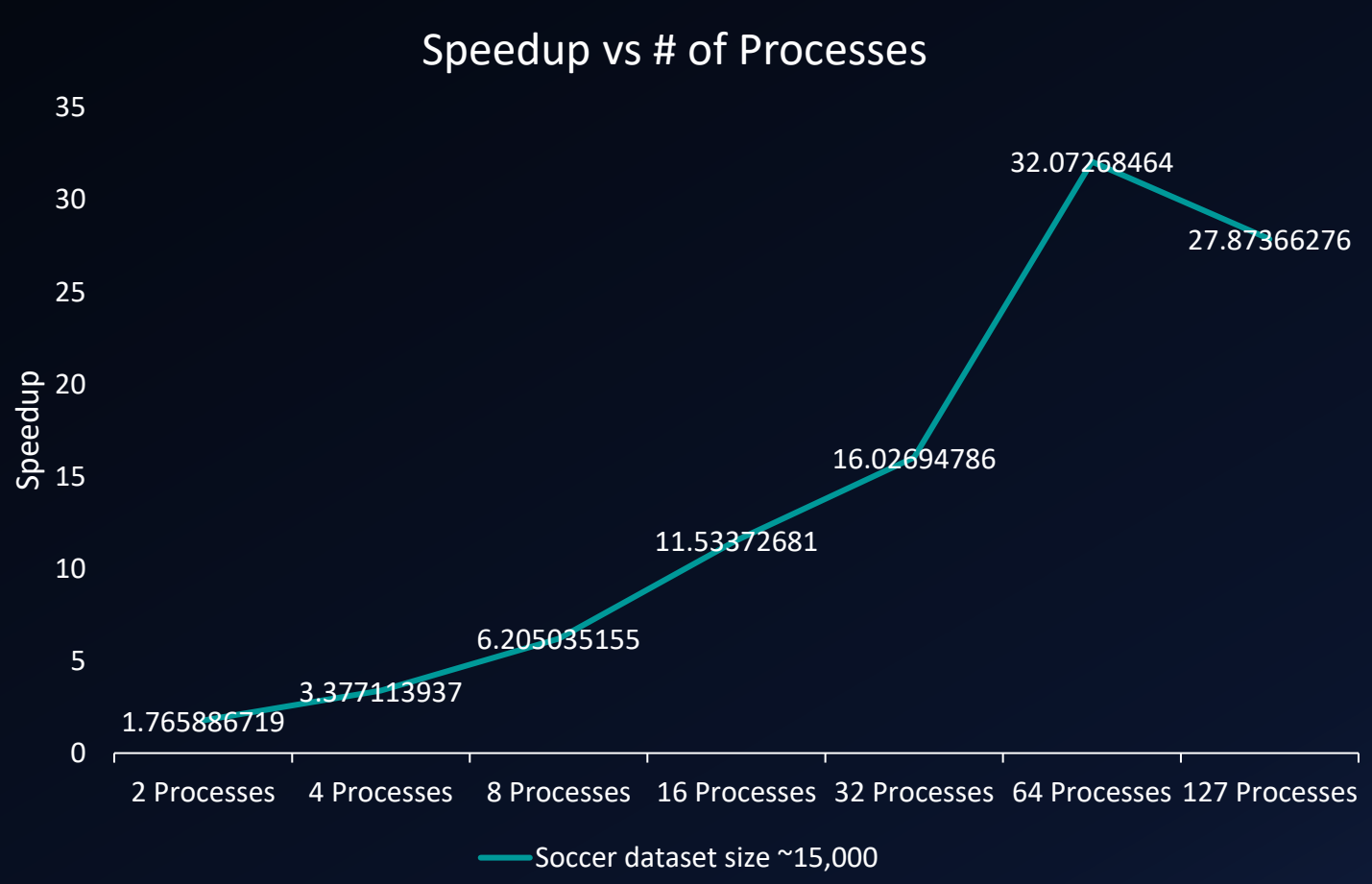
CPUs per node * number of nodes	Total number of Processes	Time (in seconds)
16 * 1	16	48.63
16 * 2	32	28.73
16 * 4	64	18.52
16 * 8	128	11.02
16 * 16	256	11.9
16 * 32	512	33.5
16 * 64	1024	112.45
16 * 127	2032 *	258.53

* as 128 nodes NA for CPU-E5-2650v2&IB, 127 used instead

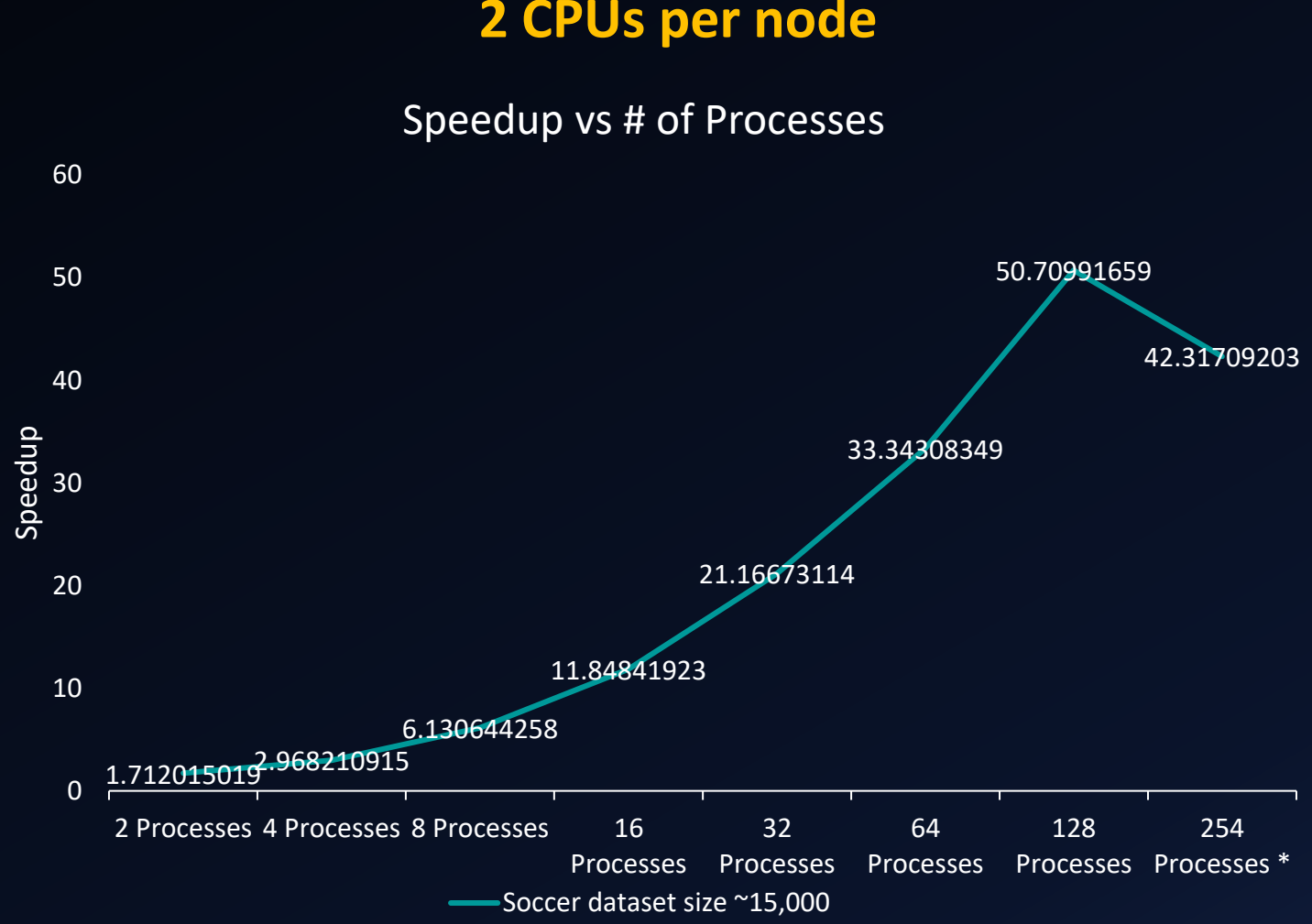


Speedup for soccer dataset

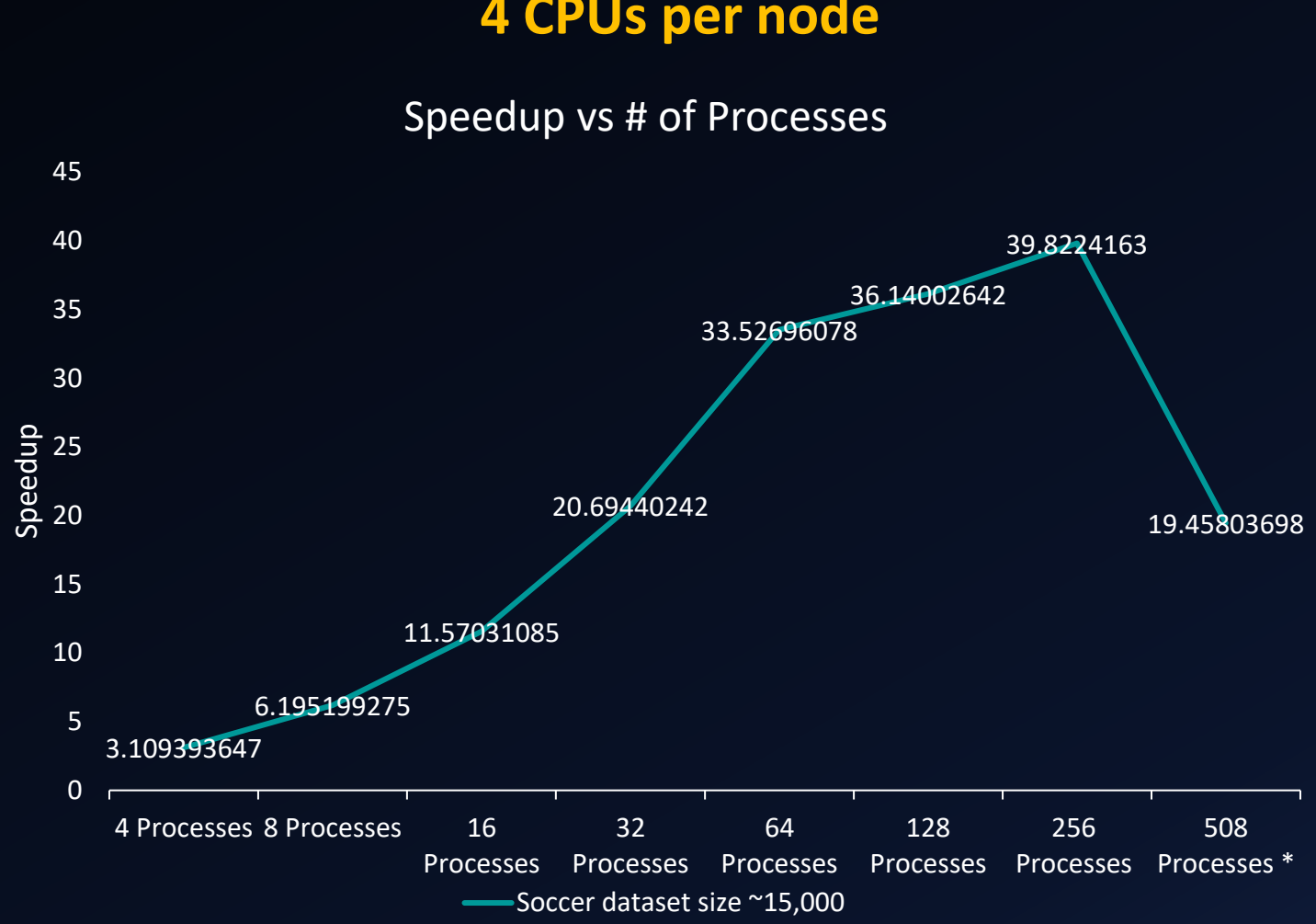
CPUs per node * number of nodes	Total number of Processes	Speedup (Sequential : 547.16 seconds)
1 * 2	2	1.76
1 * 4	4	3.37
1 * 8	8	6.20
1 * 16	16	11.53
1 * 32	32	16.02
1 * 64	64	32.07
1 * 127	127 *	27.87



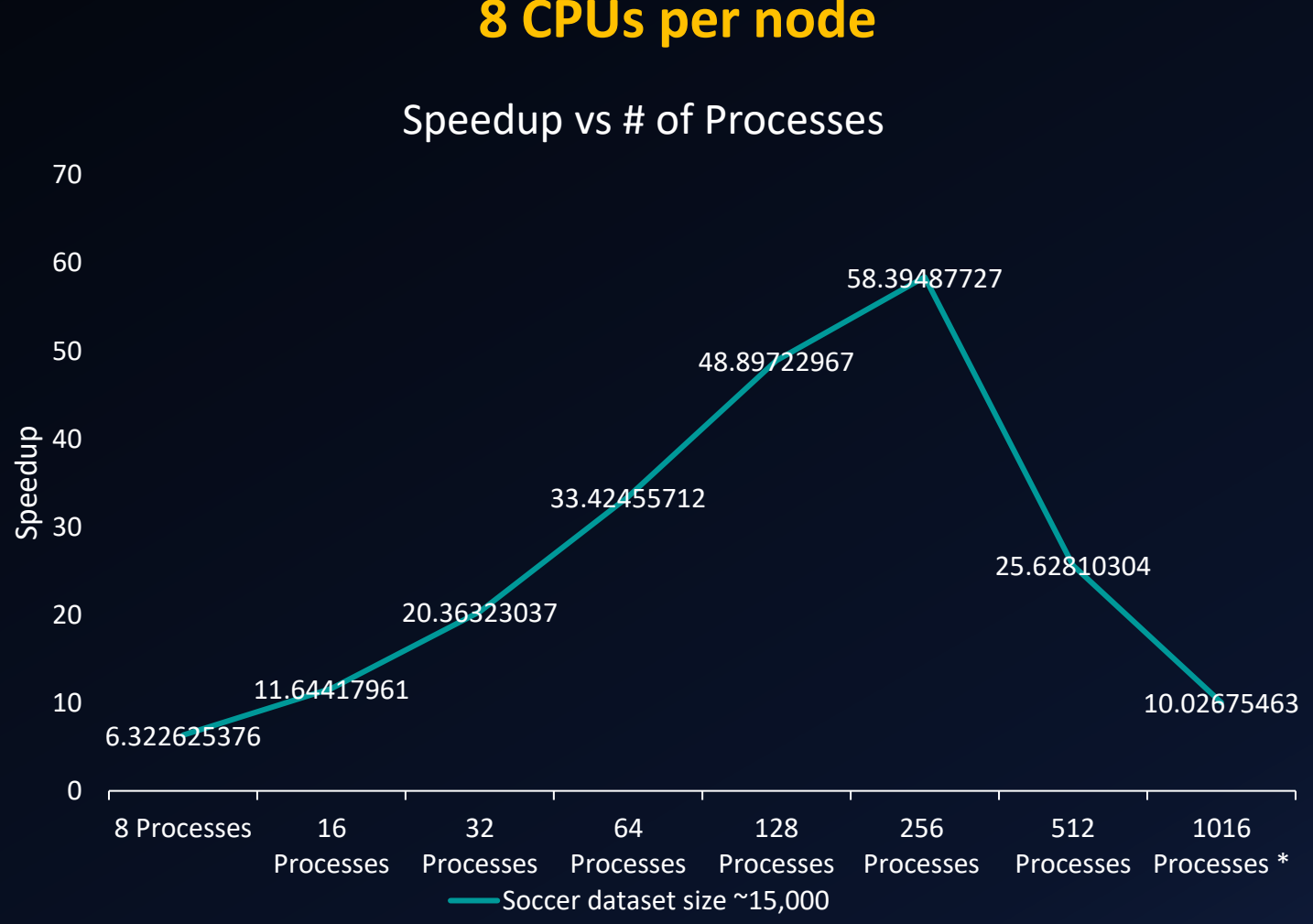
CPUs per node * number of nodes	Total number of Processes	Speedup (Sequential : 547.16 seconds)
2 * 2	2	1.71
2 * 2	4	2.96
2 * 4	8	6.13
2 * 8	16	11.84
2 * 16	32	21.16
2 * 32	64	33.34
2 * 64	128	50.70
2 * 127	254 *	42.31



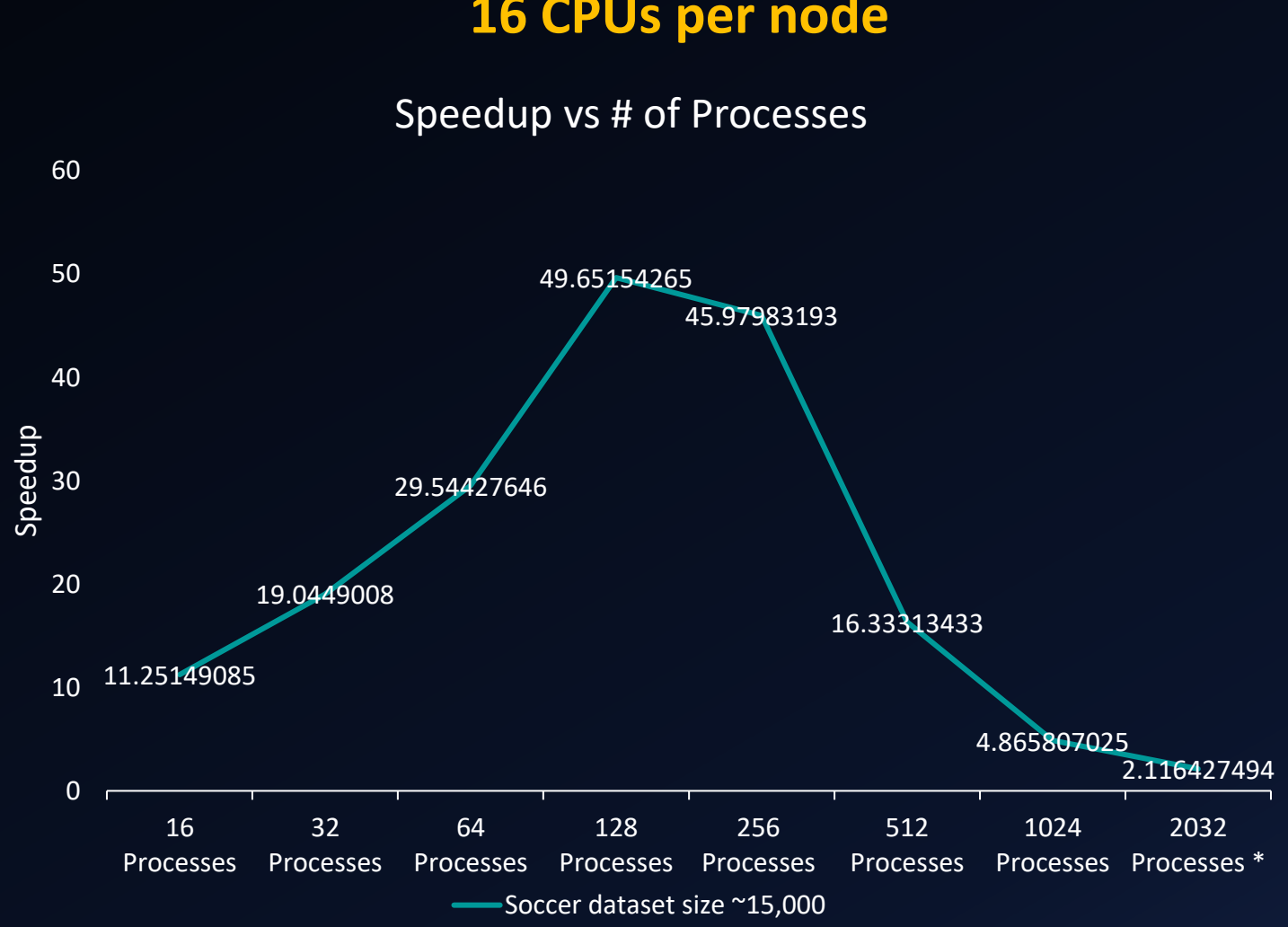
CPUs per node * number of nodes	Total number of Processes	Speedup (Sequential : 547.16 seconds)
4 * 1	4	3.10
4 * 2	8	6.19
4 * 4	16	11.57
4 * 8	32	20.69
4 * 16	64	33.52
4 * 32	128	36.14
4 * 64	256	39.82
4 * 127	508*	19.45



CPUs per node * number of nodes	Total number of Processes	Speedup (Sequential : 547.16 seconds)
8 * 1	8	6.32
8 * 2	16	11.64
8 * 4	32	20.36
8 * 8	64	33.42
8 * 16	128	48.89
8 * 32	256	58.39
8 * 64	512	25.62
8 * 127	1016 *	10.02



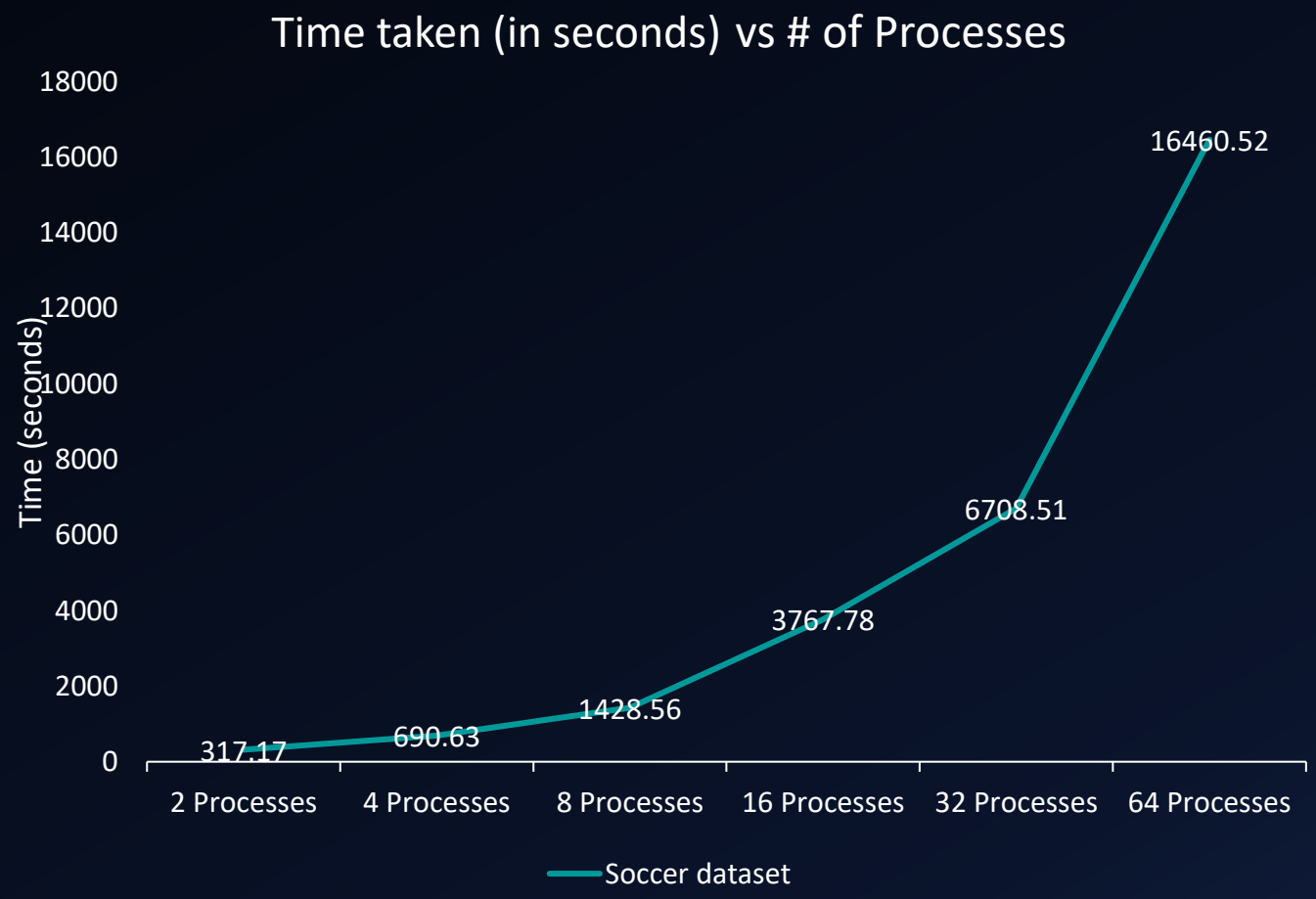
CPUs per node * number of nodes	Total number of Processes	Speedup (Sequential : 547.16 seconds)
16 * 1	16	11.251
16 * 2	32	19.04
16 * 4	64	29.54
16 * 8	128	49.65
16 * 16	256	45.97
16 * 32	512	16.33
16 * 64	1024	4.86
16 * 127	2032 *	2.11



Scaling for soccer dataset

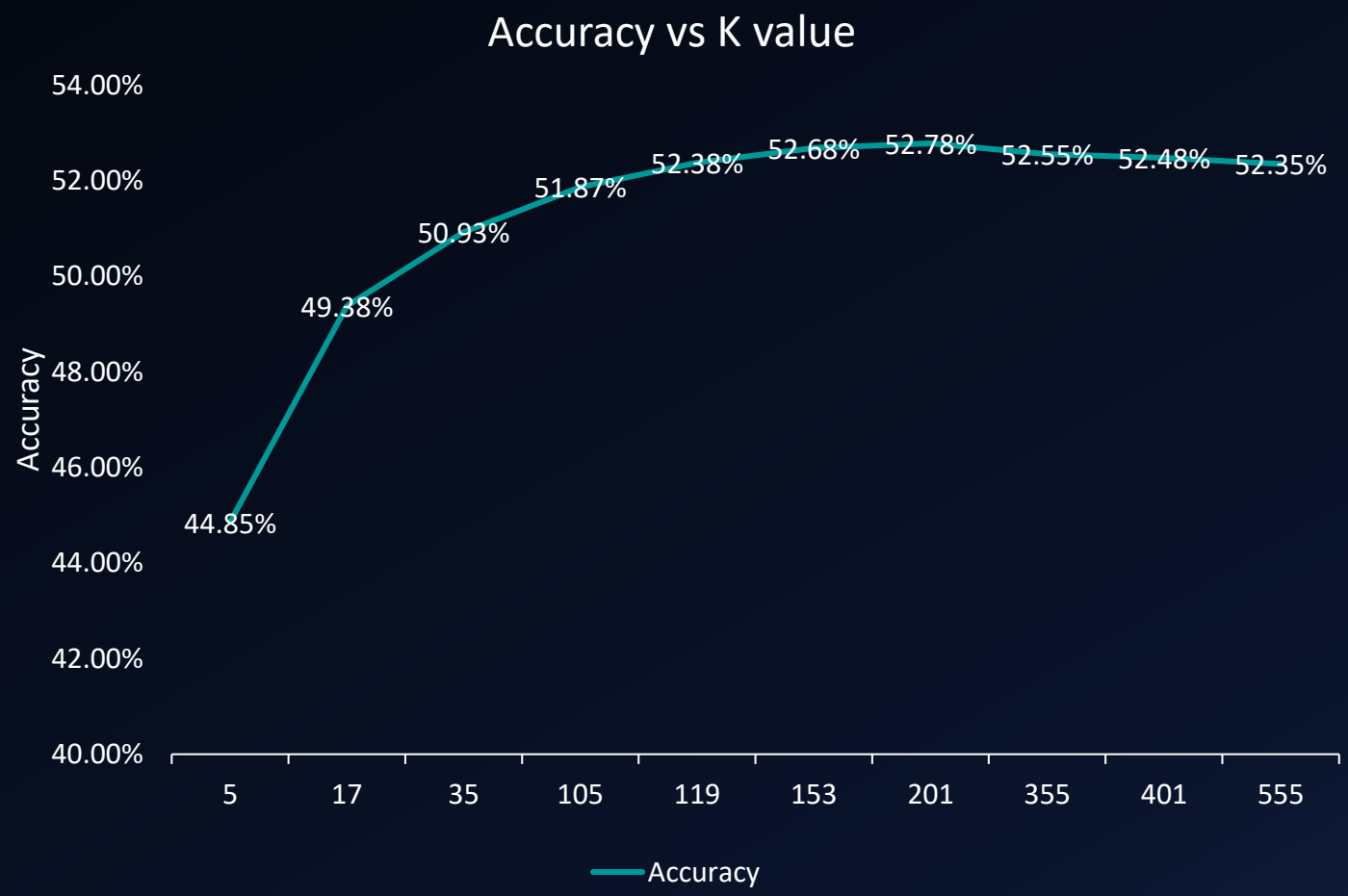
1 CPU per node
Data per process: 5508

CPUs per node * number of nodes	Total number of Processes	Total Training data	Time (in seconds)
1 * 2	2	11016	317.17
1 * 4	4	22032	690.63
1 * 8	8	44064	1428.56
1 * 16	16	88128	3767.78
1 * 32	32	176256	6708.51
1 * 64	64	352512	16460.52



K value selection

K value	Accuracy
5	**44.85%**
17	49.38%
35	50.93%
105	**51.87%**
119	52.38%
153	52.68%
201	**52.78%**
355	52.55%
401	52.48%
555	52.35%



Intended next steps

- Using Slurm scripts to scale further
- Speedup
- Apply KNN model to a real dataset
  - Soccer dataset
  - Can be used to predict match outcomes or fantasy premier leagues teams
- Selection of K value
- Weighted nearest neighbor to improve accuracy

REFERENCES

- KNN details
  1. https://www.ibm.com/topics/knn#:~:text=Next%20steps-,K%2DNearest%20Neighbors%20Algorithm,of%20an%20individual%20data%20point
  2. https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
- Sample Dataset generation
  3. https://scikit-learn.org/stable/datasets/sample_generators.html#generators-for-classification-and-clustering
- MPI/Slurm
  4. https://ubccr.freshdesk.com/support/solutions/articles/13000026245-tutorials-and-training-documents
- MPI tutorial
  5. https://github.com/mpitutorial/mpitutorial
- Soccer match datasets:
  6. https://www.kaggle.com/datasets/hugomathien/soccer (original dataset)
  7. https://www.kaggle.com/code/zavodrobotov/match-outcome-prediction-in-football-c23055/notebook (processed dataset)

THANK YOU