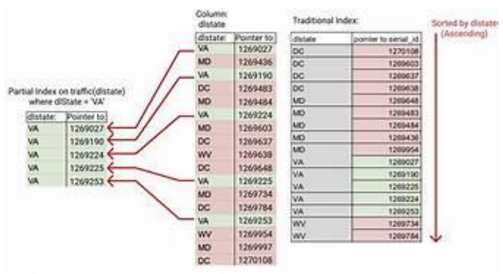
Benchlex: A Benchmark for ALEX

Alexander Song[1, 4], Clara Henzinger[2, 4], Savir Basil[3, 4], Andy Huynh[4], Aneesh Raman[4], Manos Athanassoulis[4]

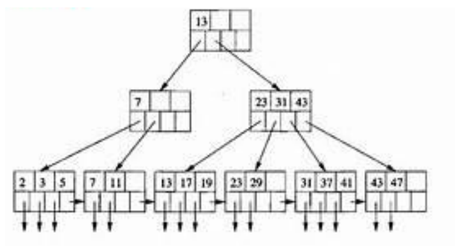
William P. Clements High School (4200 Elkins Rd, Sugar Land, TX 77479)[1] Lycée Français de Vienne (Liechtensteinstraße 37A, 1090 Vienna, Austria)[2] Sharon High School (181 Pond St, Sharon, MA 02067)[3] Boston University (665 Commonwealth Ave, Boston, MA 02215)[4]

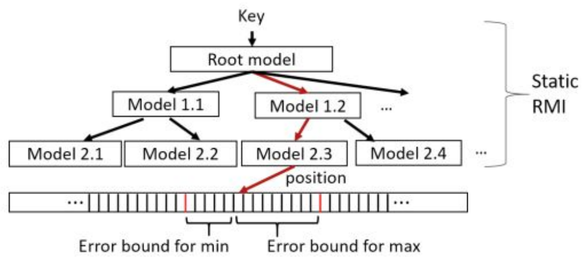
Traditional Indexes

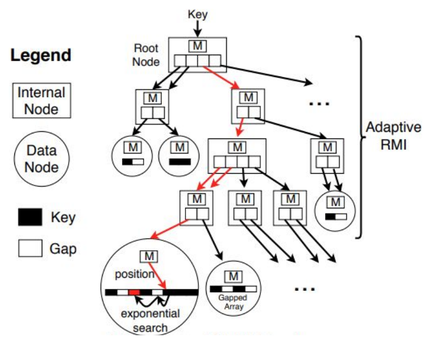


Indexes are ubiquitous in society to store important data.

Learned Indexes







Learned Indexing replaces conventional indices.

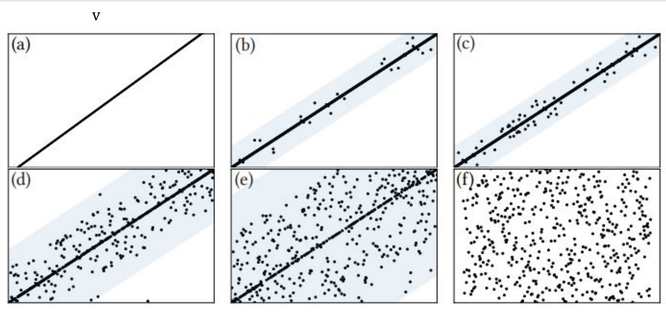
The recursive model index guides queries to other machine learning models and then the result.

ALEX: An Adaptable Learned Index, accommodates for updates.

Sortedness

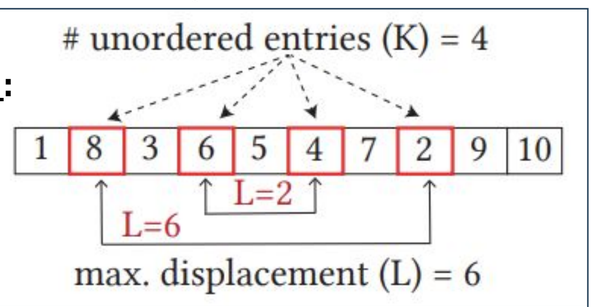
How sorted is the data? Sortedness - how scrambled the dataset is

Real world datasets are often nearly sorted



Key vs. Position

BoDs Sortedness Benchmark :
K - total displaced keys
L - maximum displacement of any key



Benchmarking ALEX with BoDs

Experimental Process

Generate a dataset with BoDs → **Build a C++ Benchmark in ALEX to read the BoDs dataset** → **Plot Performance using Python MatPlotLib**

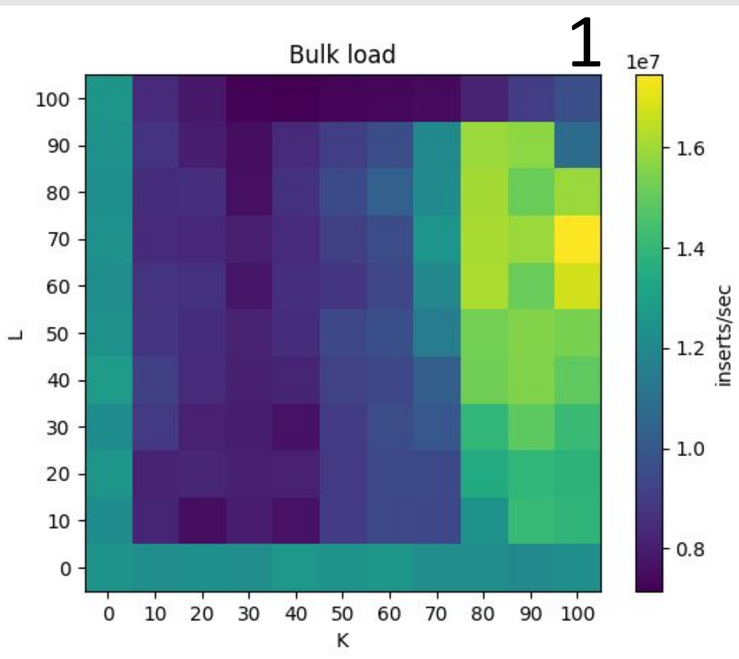
Experimental Values

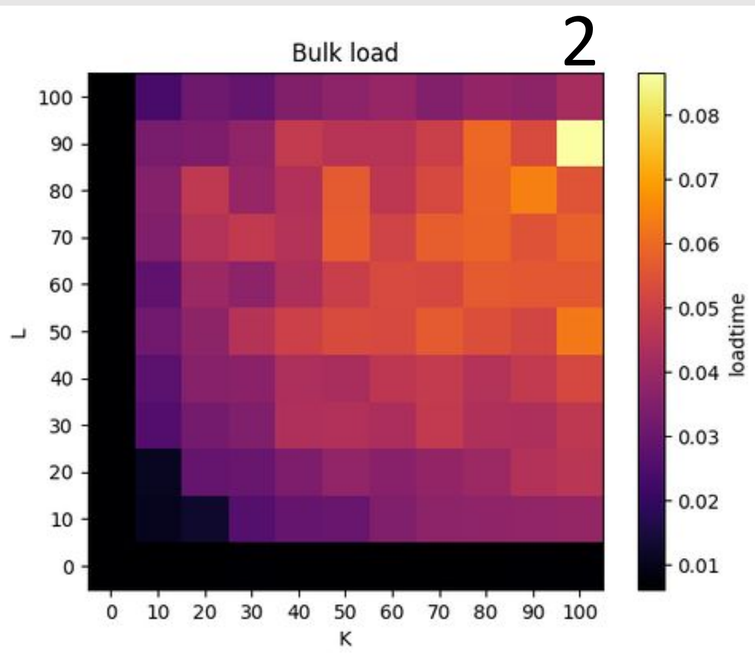
Sortedness : Values from 0-100% on both K and L **[graphs below]**

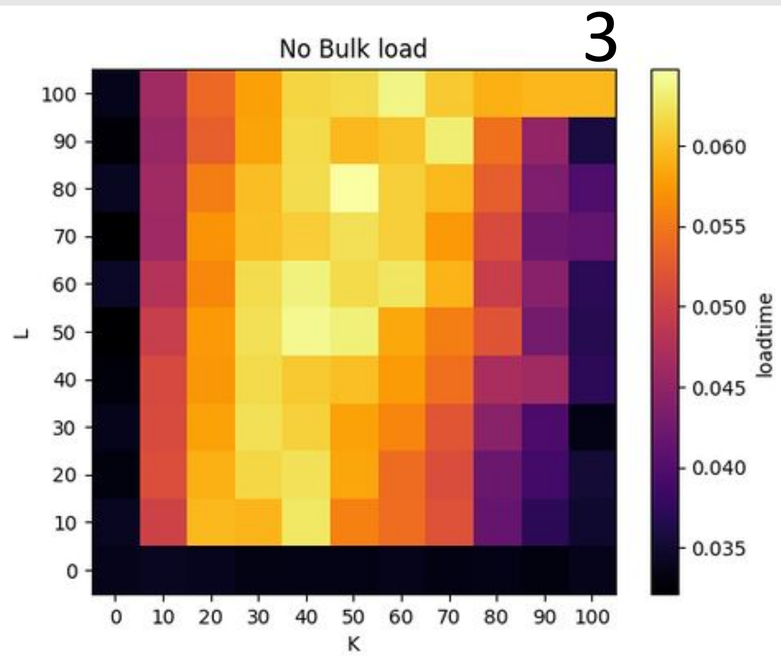
Data Set : On different sizes of data.

Bulk Loading : Pre-sorted and inserted in bulk instead of individually. Without bulk loading, individual inserts are learned.

Results







Observations

Mostly Unsorted data is quicker → **Machine Learning performs better on random data**

Completely sorted datasets performed well, as expected. (K=0 or L=0)

Large K, Small L is locally unsorted, so it is as fast as fully sorted in graph 3.

Bulk loading has a much quicker overall load time than individual inserts.

Conclusion

ALEX learns better on randomized data as shown in an increase of performance with greater sortedness.

Without bulk loading (graph 3), it is easy to learn locally unsorted data, and therefore, load time is almost as quick as in fully sorted data.

Future Inquiry

How does ALEX's learning model interact with bulk loaded datasets and single insert datasets?

How can this be adapted to take advantage of near sorted data which is more relevant?

References

Kraska, T., Beutel, A., Chi, E. H., Dean, J., & Polyzotis, N. (2018, May). The case for learned index structures. In *Proceedings of the 2018 international conference on management of data* (pp. 489-504).

Ding, J., Minhas, U. F., Yu, J., Wang, C., Do, J., Li, Y., ... & Kraska, T. (2020, June). ALEX: an updatable adaptive learned index. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data* (pp. 969-984).

Raman, A., Sarkar, S., Olma, M., & Athanassoulis, M. (2023). Indexing for Near-Sorted Data. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*.

Raman, A., Karatsenidis, K., Sarkar, S., Olma, M., & Athanassoulis, M. (2022, September). BoDS: A Benchmark on Data Sortedness. In *Technology Conference on Performance Evaluation and Benchmarking* (pp. 17-32). Cham: Springer Nature Switzerland.

Acknowledgements

We would like to thank Andy Huynh, Aneesh Raman, and Manos Athanassoulis for teaching and preparing us, as well as serving as a mentor and a friend. I would also like to thank my lab partners for discussion and support throughout the project.

https://github.com/BU-DiSC/BenchLex.git