Machine Learning For Databases: Learned Indexes

- Aryan Mittal
- Ansel Erol
- Sarah Chae
- Edward Chen
- Adithya Peruvemba

Georgia Tech

Introduction

Traditional Indexes

- Traditional indexes accelerate data retrieval
  - Enable rapid search and access to specific data without scanning entire tables
  - Improve query performance, especially in large datasets
  - Implemented with B-Trees, B+ Trees, LSM Trees, Hash Indexes, R Trees, etc.

- Learned indexes use ML to predict data locations

Why Learned Indexes?

- Faster Lookups
  - Make searches faster than traditional indexing methods
  - Especially for read-heavy workloads
- Space Efficiency
  - Use less memory than traditional indexes
  - Especially when the dataset follows a predictable distribution
- Adaptability
  - Dynamically adjust to different data distributions
  - Improve performance on specific workloads

Learned Index Basics

1. Training a Model
   - ML model learns data distribution
   - Builds mapping from keys to their approximate positions in storage/memory
2. Making Predictions
   - When a query searches for a key, model predicts its approximate location
3. Refinement
   - System refines prediction using localized search methods
   - E.g., binary search or small secondary indexes

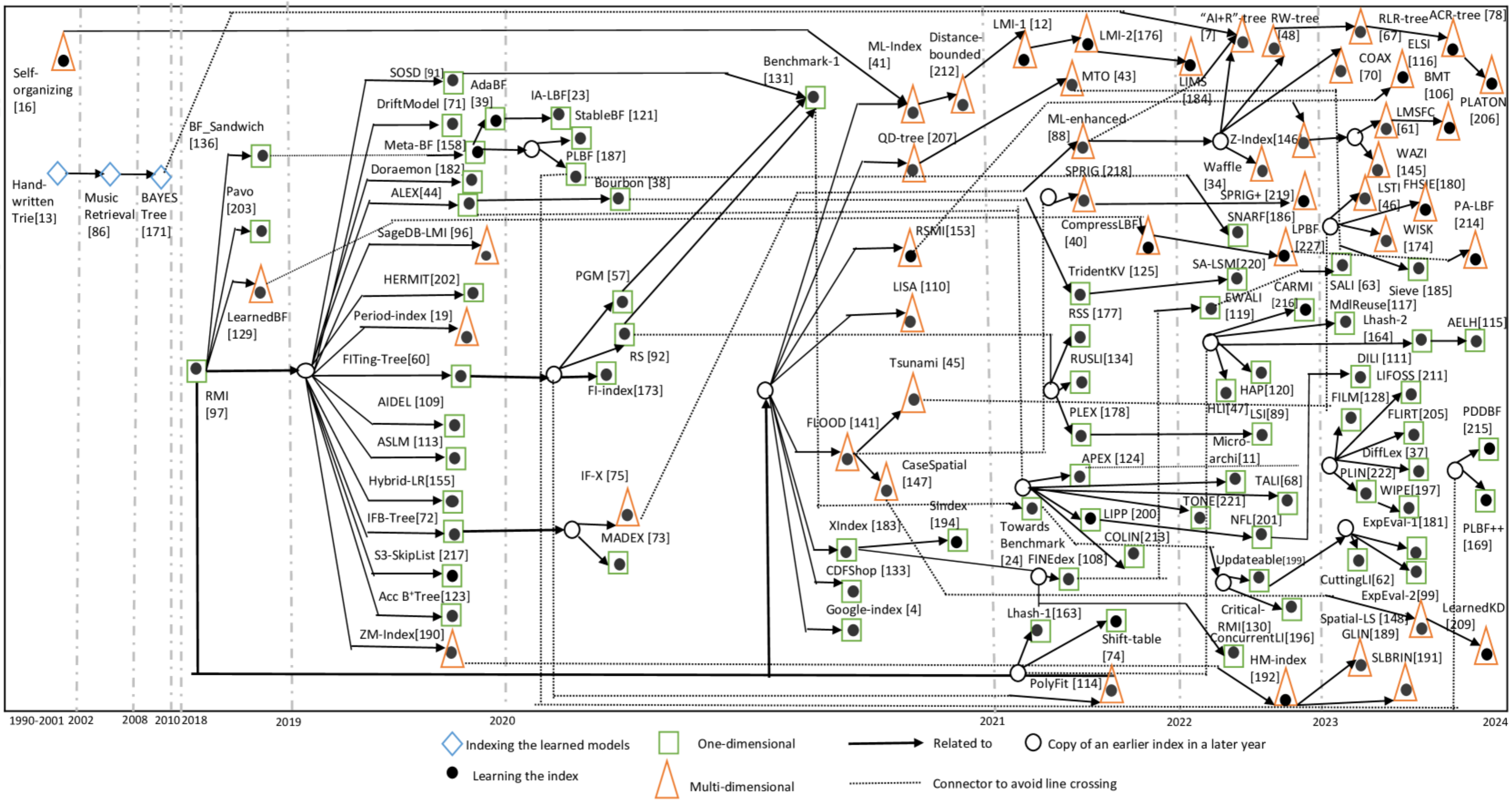
⇒ Hierarchical Structure

- Learned indexes can be organized in layers
- Top-level models predict which low-level models should be used for precision
- Similar to B-Trees

This Presentation

- Products Overview
- Technical Details
- Sample Applications
- Market Analysis
- Future Trends

Products Overview



https://arxiv.org/html/2403.06456v1 (Purdue, 2024)

Types of "Products"

Large-Scale Distributed Systems

- Google BigTable

Dynamic/Updatable Indices

- ALEX
- LIPP
- Tsunami

Hybrid

- PGM Index
- RMI
- RadixSpline

Auxiliary Uses

- Learned Bloom Filters
- CDF + RMI joins

Benchmarking

- SoSD

1D, Space-efficient Indices

- Recursive Model Index (RMI):[MIT]
    - First proposed model
    - Hierarchically approximates position of keys with sorted data instead of using B-trees
    - In joins, 2-3x less comparisons than standard equijoin algorithm
- PGM (Piecewise geometric) Index[U of Pisa, ICML]
    - Splines instead of lines
    - Automatically finds optimal number of splines given error tolerance
- Radix Splines[MIT]
    - Improvement to PGM that reduces complexity of index building
    - Higher performance on range queries

Georgia Tech

Dynamic Learned Index

- Traditional learned indices don't adapt to inserts/updates

- ALEX (Adaptive Learned Index) [Microsoft Research]
  - Expands dynamically while maintaining lookup efficiency
  - On insert, predicts leaf node of next child and splits if full (retraining the node)

- LIPP (Learned Index with Precise Positioning)[Tsinghua]
  - Optimized for workloads that are write-heavy instead of read heavy
  - On insert, predicts leaf node and adds a new node if full – faster

Georgia Tech

Google Bigtable

- Google BigTable (2005) is a NoSQL service in GCP, supporting a range of other Google products
- Integrated learned indices to augment traditional indexing strategies in 2020
- Performance Gains:
  - Optimized for large-scale disk-based storage
  - ML models reduce size while maintaining lookup speed – less IOs for index blocks
- First (and only) large scale, real-world deployment of learned indices in a distributed setting
  - Insight – for distributed, large scale deployment, savings on index size and simpler prefetching matter more than faster lookup

Specialized, Multi-Dimensional, etc.

- Flood and Tsunami [MIT]
  - Works in multiple-dimensions and adapts to dataset characteristics
  - Outperforms R-Trees, the standard approach in multiple dimensions

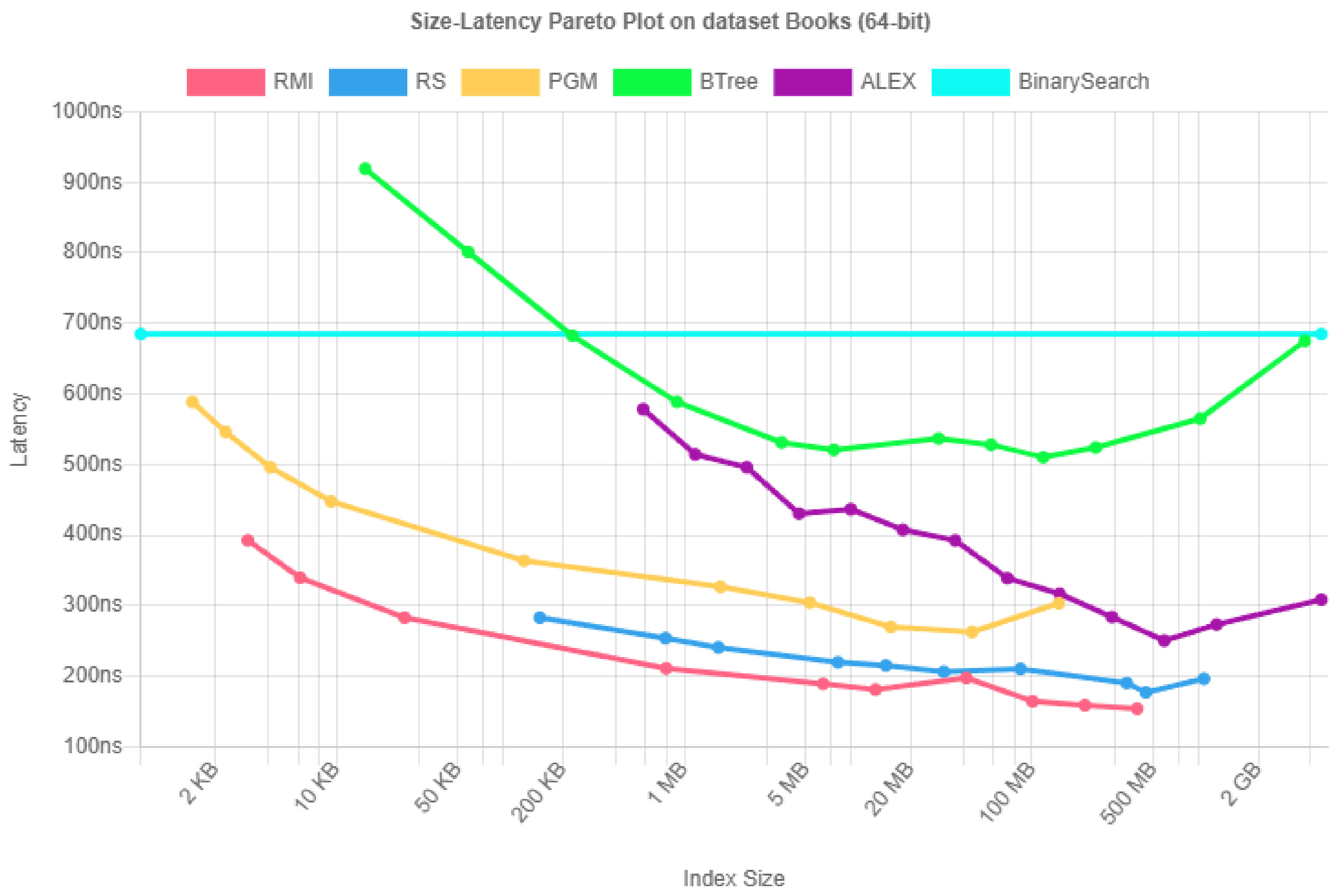
- Learned Bloom Filters:
  - Existence queries with machine learning
  - Achieves 100% recall with higher precision than the traditional algorithm

Applications

- Geographic Databases, spatial queries, in high dimensions

Comparison

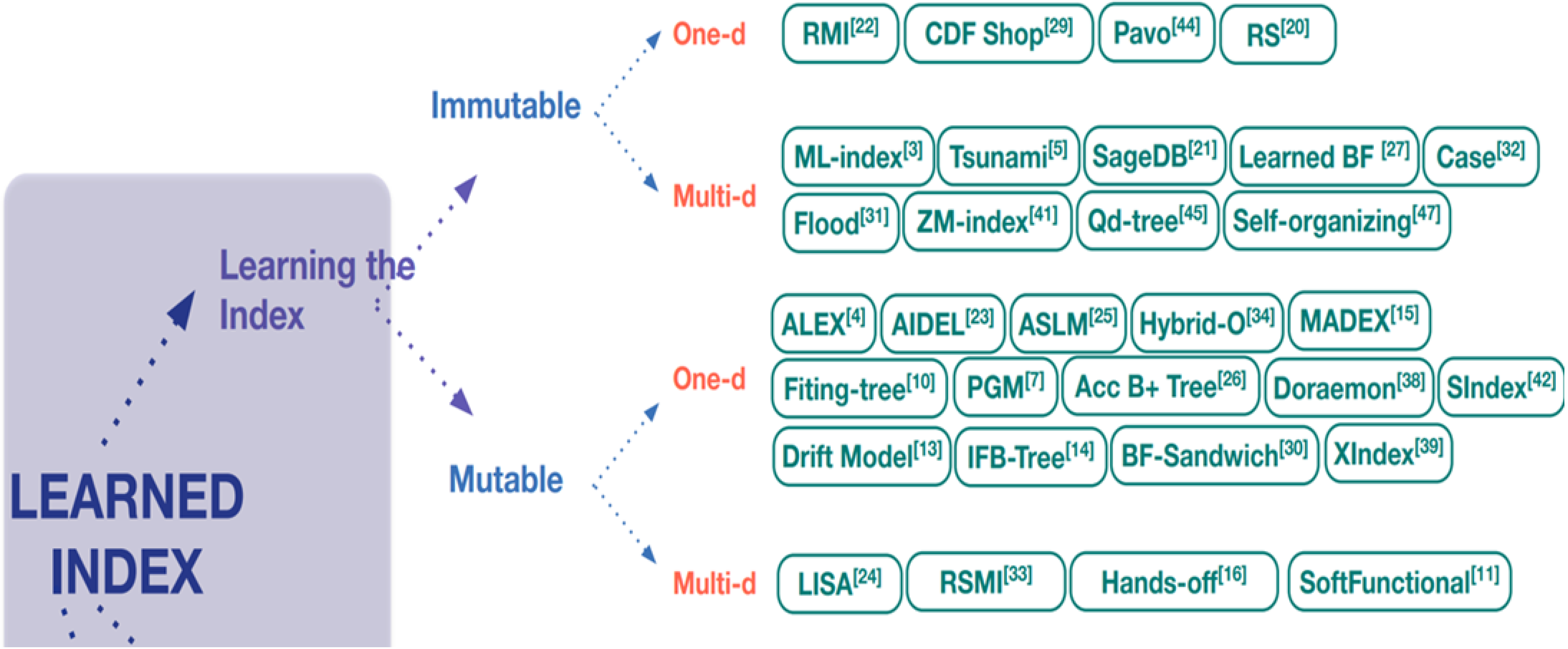
Product	Category	Dynamic Support	Space Efficiency	Real-World Deployment	Primary Use Case
Google Bigtable Learned Index	Distributed / Enterprise	Yes	High	Yes (integrated in Google Bigtable)	Large-scale, disk-based storage & analytics
ALEX (Adaptive Learned Index)	Dynamic / Updatable	Yes	Med-High	Prototype / Research	KV store w/ frequent updates
PGM (Piece-wise Geometric)	Space-Efficient / Immutable	No	Very High	Prototype / Research	Read-only, analytical workload
RadixSpline (RS)	Space-Efficient / Immutable	No	Very High	Prototype / Research	Read-only, in-memory + range queries
Recursive Model Index (RMI)	Pure learned / Immutable	No	Medium	Prototype / Research	Read only point + range queries



https://learnedsystems.github.io/SOSDLeaderboard/leaderboard/ [MIT]

Technical Details

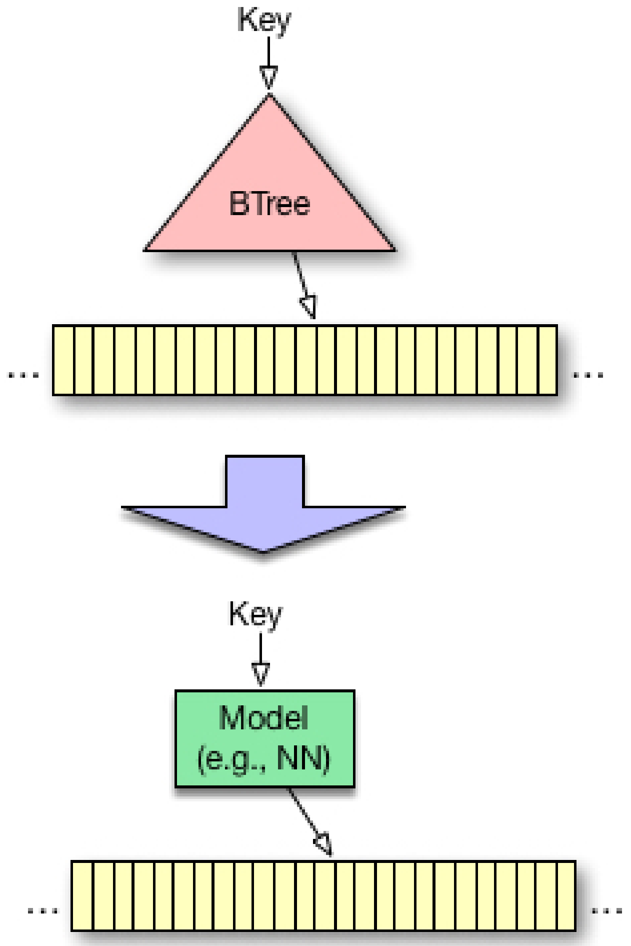
Taxonomy of Learned Indexes



[3] Mamun, A. A., Wu, H., & Aref, W. G. (n.d.). A tutorial on learned multi-dimensional indexes. https://www.cs.purdue.edu/homes/aref/LMDI2020/LMDI_Tutorial_SIGSpatial2020.pdf

The Case of Learned Index Structure [SIGMOID'18]

- Introduced the idea that "Indexes are models"
- Replace traditional database indexes by learned models
- Approximate the Cumulative Distribution Function (CDF) of the underlying (sorted) data
- Proposed Recursive Model Index (RMI), a multi-stage ML model
- Combine simpler ML models
    - The first stage model will make an initial prediction of the CDF for a specific key
    - The next stage models will be selected to refine this initial prediction
- Proposed Learned Index Structures: Range Index, Point Index, and Existence Index



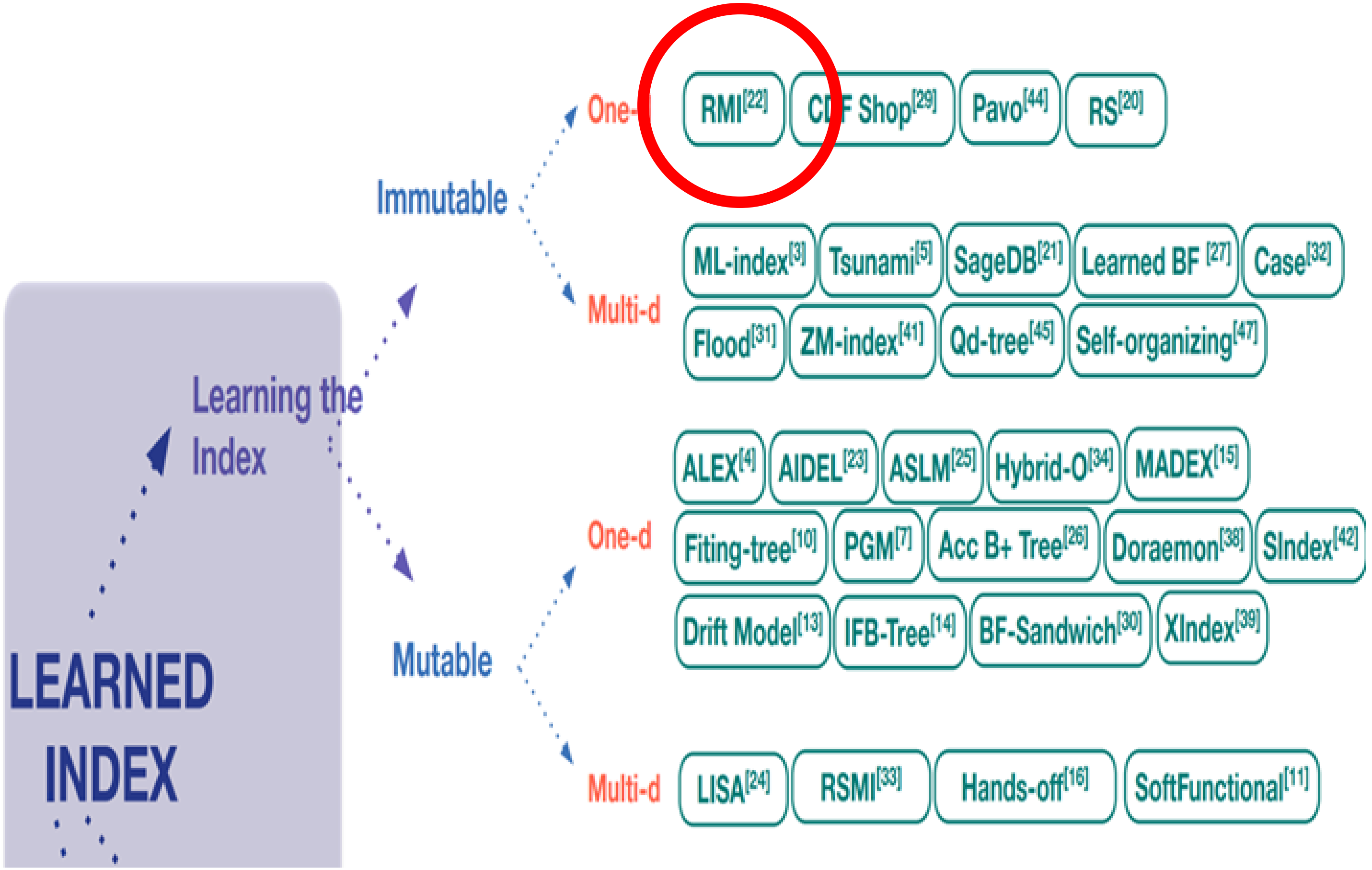
[2] Kraska, T., Beutel, A., Chi, E. H., Dean, J., & Polyzotis, N. (2018). The Case for Learned Index Structures. *arXiv:1712.01208 [cs.DB]*.

Georgia Tech

The Case of Learned Index Structure [SIGMOID'18]

- Limitations:
  - Focus on in-memory read-only workloads
  - The structure of RMI is static
  - Does not support updates (e.g., insertion, deletion)

- This leads to the development of Dynamic Learned Indexes



[2] Kraska, T., Beutel, A., Chi, E. H., Dean, J., & Polyzotis, N. (2018). The Case for Learned Index Structures. *arXiv:1712.01208 [cs.DB]*.

Georgia Tech

Immutable vs. Mutable Learned Indexes

- **Challenges of Updates:**
  - **Training Times**: Learning indexes require significant time to train
  - **Data Changes**: New data necessitates retraining as it alters the data order
- **Classification based on Update Support:**

1. **Immutable Learned Indexes:**
   - **Support:** Does not support inserts, updates, or deletes
   - Once built, the structure remains static
   - Best suited for stable datasets where changes are infrequent
2. **Mutable Learned Indexes:**
   - **Support:** Allows for inserts, updates, and deletes
   - Dynamic and adaptable to changing data
   - Essential for environment where data frequently changes

[3] Mamun, A. A., Wu, H., & Aref, W. G. (n.d.). A tutorial on learned multi-dimensional indexes. https://www.cs.purdue.edu/homes/aref/LMDI2020/LMDI_Tutorial_SIGSpatial2020.pdf

ALEX: An Updatable Adaptive Learned Index [SIGMOID'20]

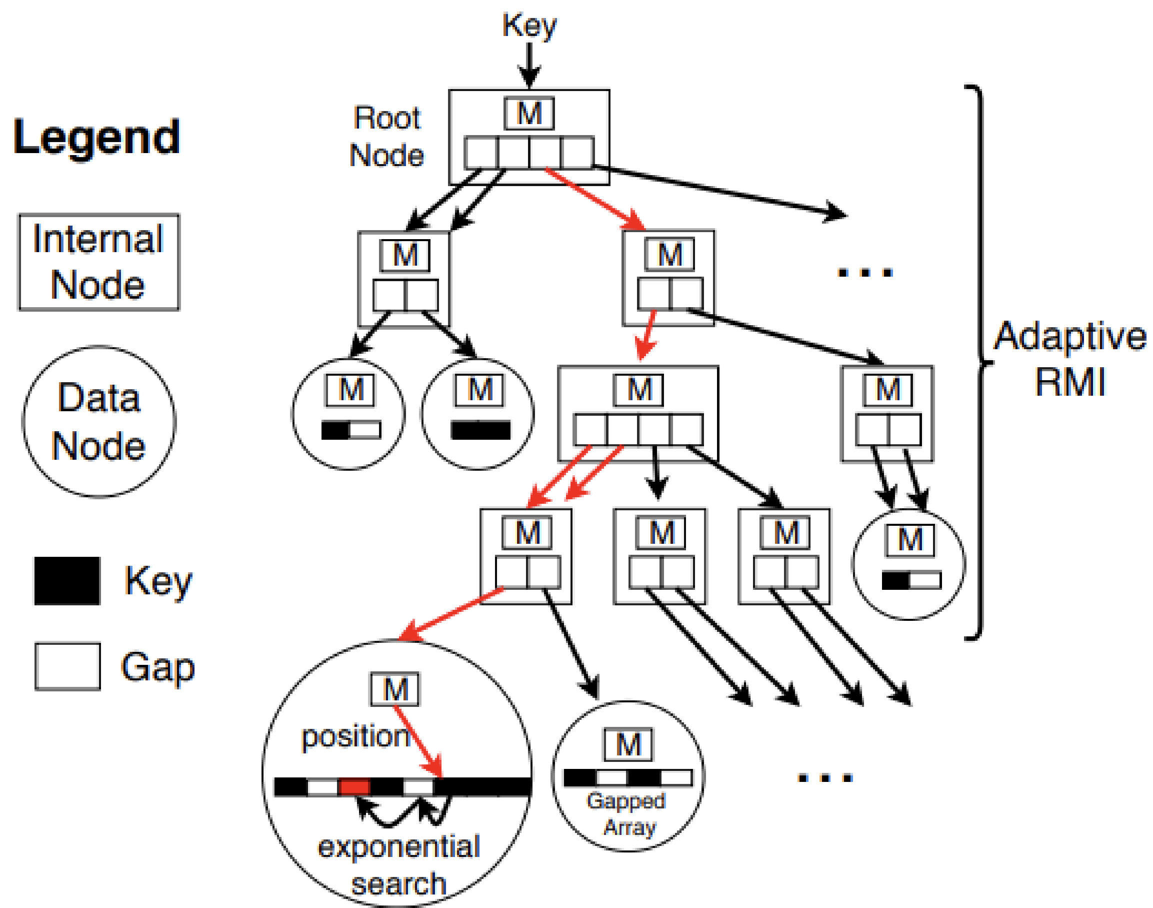


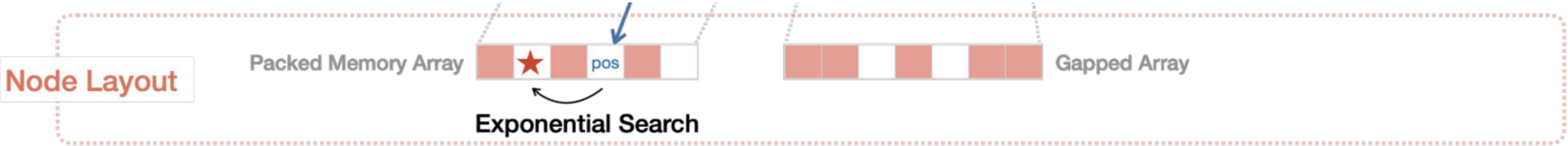
Figure 2: ALEX Design

- Dynamic, updatable learned index to handle dynamic workload
- Adaptive RMI as Model Hierarchy
- Linear Regression Models as Node
- Gapped Array or Packed Memory Array as Node Layout

[1] Ding, J., Minhas, U. F., Yu, J., Wang, C., Do, J., Li, Y., Zhang, H., Chandramouli, B., Gehrke, J., Kossmann, D., Lomet, D., & Kraska, T. (2020). ALEX: An Updatable Adaptive Learned Index. In Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (SIGMOD'20), June 14–19, 2020, Portland, OR, USA. ACM, New York, NY, USA. https://doi.org/10.1145/3318464.3389711

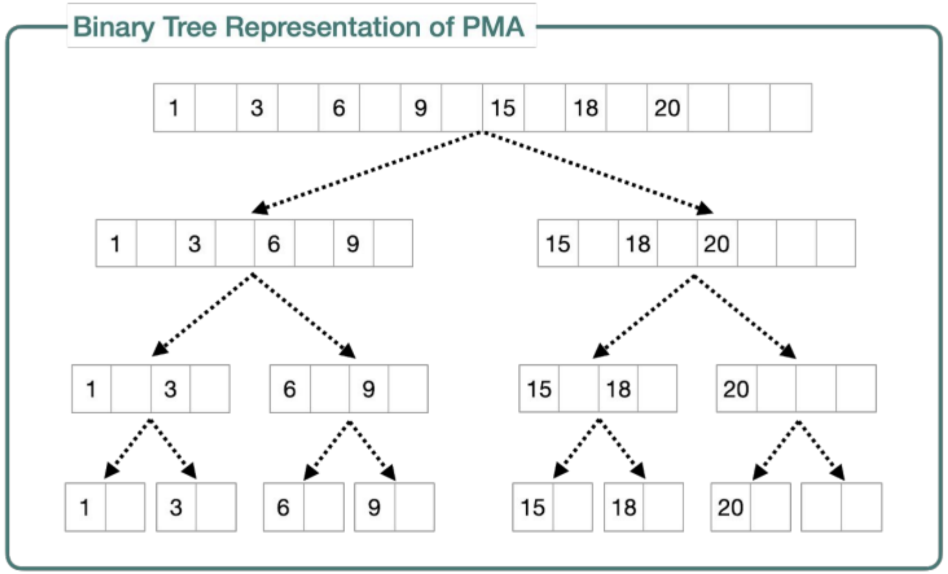
Georgia Tech

ALEX: An Updatable Adaptive Learned Index [SIGMOID'20]



1 Gapped Array (GA)

2 Packed Memory Array (PMA)



[3] Mamun, A. A., Wu, H., & Aref, W. G. (n.d.). A tutorial on learned multi-dimensional indexes. https://www.cs.purdue.edu/homes/aref/LMDI2020/LMDI_Tutorial_SIGSpatial2020.pdf

Sample Applications

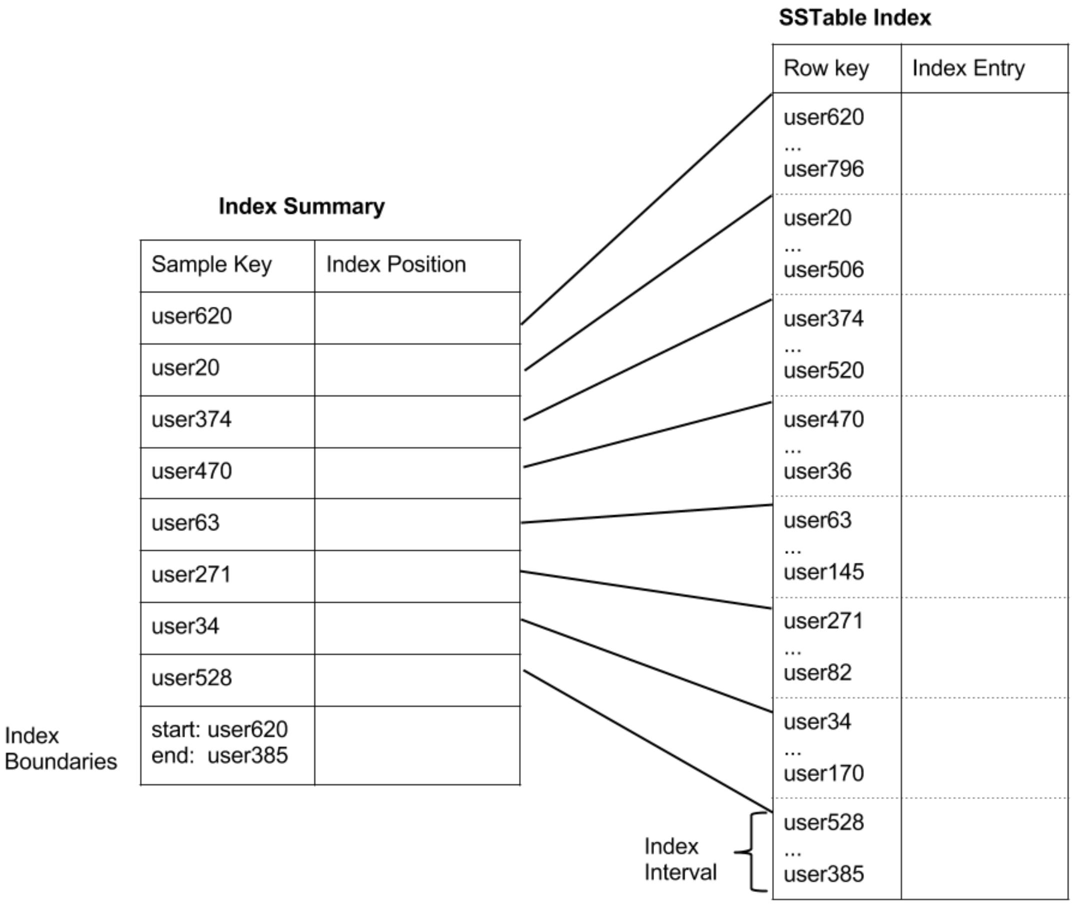
Learned Indexes for a Google-scale Disk-based Database

- Challenge: Can learned indexes be implemented in a real-world database system?
- Abu-Libdeh et al. demonstrate that the learned index can be integrated into Google Bigtable
- Improves **end-to-end latency** and **throughput**

[4] H. Abu-Libdeh, D. Altınbüken, A. Beutel, E. H. Chi, L. Doshi, T. Kraska, X. (S.) Li, A. Ly, and C. Olston, "Learned Indexes for a Google-scale Disk-based Database," Workshop on ML for Systems, Vancouver, Canada, December 2020.

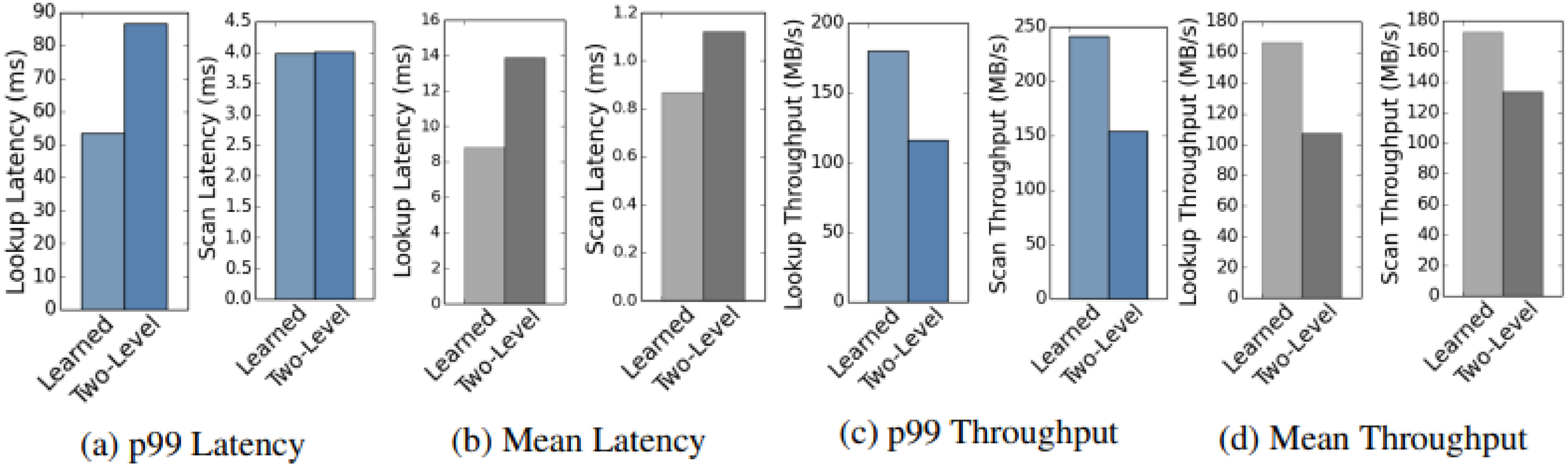
Learned Indexes for a Google-scale Disk-based Database

- Google Bigtable uses a **SSTable** with key-value pairs stored in blocks
- Use model to define which records are stored in which block
  - Train a model to predict number of bytes a record stores
  - Use **linear regression** model (strength in simplicity)



[4] H. Abu-Libdeh, D. Altınbüken, A. Beutel, E. H. Chi, L. Doshi, T. Kraska, X. (S.) Li, A. Ly, and C. Olston, “Learned Indexes for a Google-scale Disk-based Database,” Workshop on ML for Systems, Vancouver, Canada, December 2020.

Learned Indexes for a Google-scale Disk-based Database



[4] H. Abu-Libdeh, D. Altınbüken, A. Beutel, E. H. Chi, L. Doshi, T. Kraska, X. (S.) Li, A. Ly, and C. Olston, "Learned Indexes for a Google-scale Disk-based Database," Workshop on ML for Systems, Vancouver, Canada, December 2020.

Learned Indexes for a Google-scale Disk-based Database

- Positive impacts:
  - Significant improvement in read performance compared to two-level index (Bigtable default)
  - Reduces CPU resource usage and block accesses
  - Overall cascading benefits from smaller size and simple usage
- Limitations:
  - Training overhead and maintenance
  - Is not compatible with dynamic operations, e.g. write/updates

Google BigTable

[4] H. Abu-Libdeh, D. Altınbüken, A. Beutel, E. H. Chi, L. Doshi, T. Kraska, X. (S.) Li, A. Ly, and C. Olston, "Learned Indexes for a Google-scale Disk-based Database," Workshop on ML for Systems, Vancouver, Canada, December 2020.

Market Analysis

Current Market Positioning

- Major players in learned indexing
    - Google (Bigtable)
    - Microsoft (ALEX)
    - Academics (e.g., Radix Spline, PGM)
- Traditional B-Trees and Hash Indexes still dominate enterprise solutions
- Learned indexes show promise in read-heavy and analytical workloads
    - Business intelligence reporting
    - Large scale search

Microsoft

Georgia Tech

Performance & Revenue Impact

- Time Reduction: 1.5—3x query performance improvements
  - Microsoft ALEX reports 1.5—3x faster point lookups compared to B-trees in experiments [1]
  - Google reports 1.5—2.2x performance improvements over B+ trees [2]

- Infrastructure Cost Reduction
  - FLOOD reports 15-45% storage efficiency improvements [7]

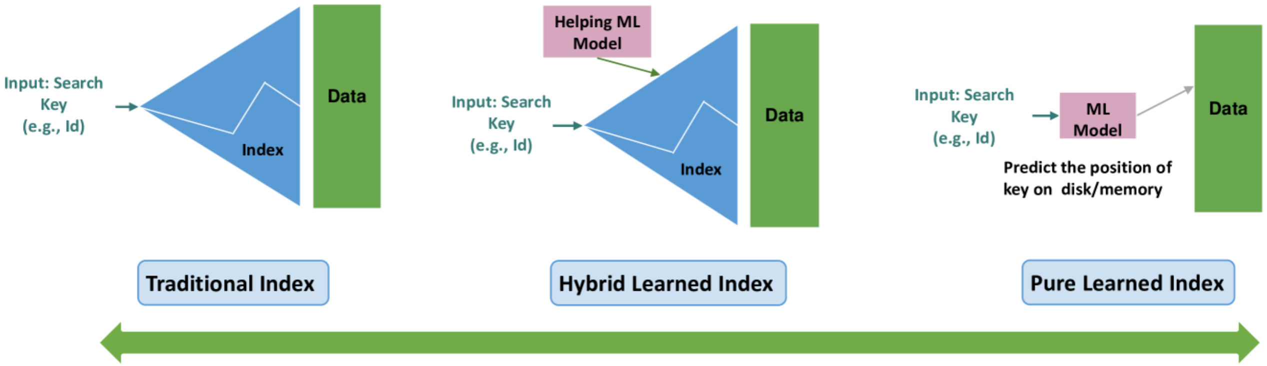
- Lower Total Cost of Ownership
  - Reduced hardware and energy requirements

Future Trends

Georgia Tech

Advancements in Mutable and Hybrid Learned Indexes

1. **Evolution of Dynamic and Fully Mutable Learned Indexes**
   - **Trend: Increasing focus on fully mutable learned indexes for real-time updates.**
   - **Key Innovation: ALEX and selective retraining techniques.**
   - **Future Direction: Incremental learning and error correction to minimize retraining.**



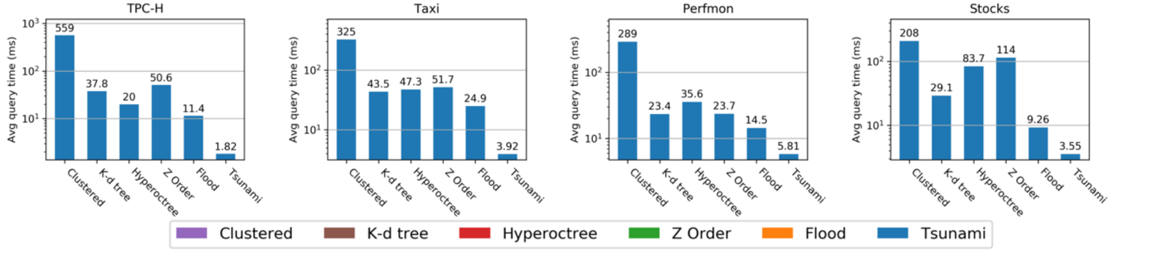
2. **Hybrid Learned Indexes with Predictive Optimization**
   - **Trend: Integration of B-trees, hash-based indexes, and learned models for efficiency.**
   - **Key Innovation: Hybrid RMI and cache-aware strategies to reduce CPU cache misses.**
   - **Future Direction: Adaptive hybrid indexes that adjust to real-time workloads.**

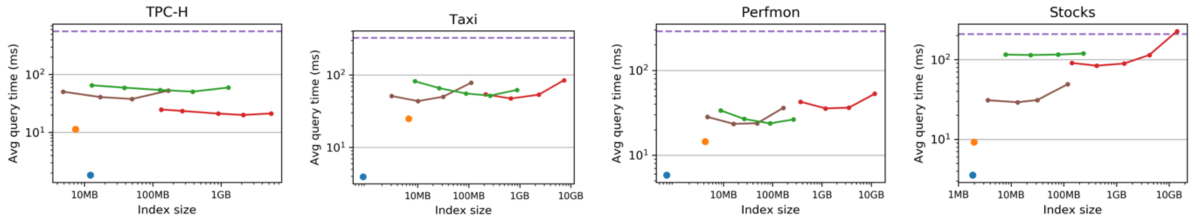
[2] Kraska, T., Beutel, A., Chi, E. H., Dean, J., & Polyzotis, N. (2018). The Case for Learned Index Structures. Proceedings of the 2018 ACM SIGMOD International Conference on Management of Data. https://doi.org/10.1145/3183713.3196909

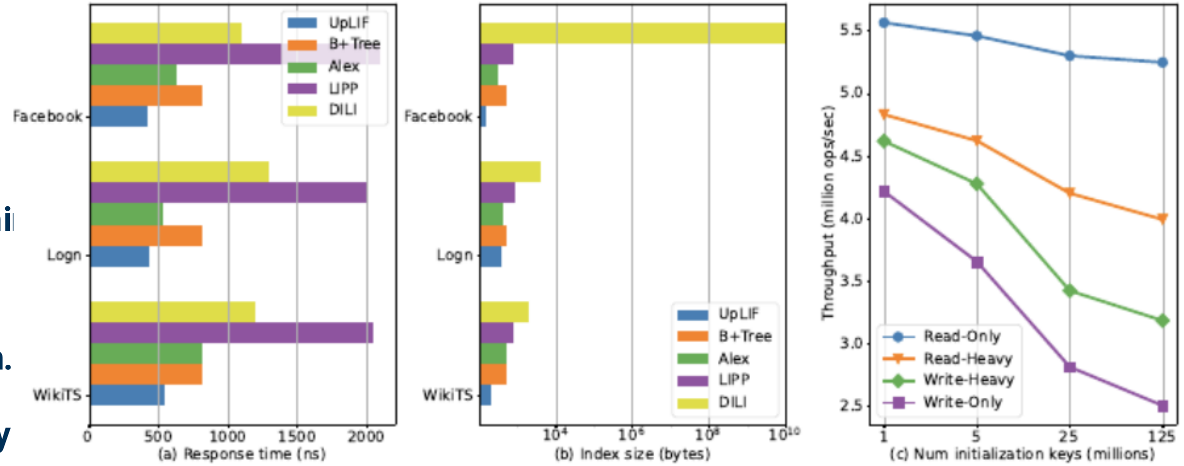
Georgia Tech

Multi-Dimensional and Self-Optimizing Indexes

1. **Advancements in Multi-Dimensional and Spatial Learnec Indexes**
   - **Trend: Expansion into GIS, IoT, and spatial databases.**
   - **Key Innovation: FLOOD and Tsunami for adaptive grid-based indexing.**
   - **Future Direction: Correlation-aware and space-filling curve-based indexes.**
2. **Self-Optimizing Learned Indexes with AI and Reinforcement Learning**
   - **Trend: Development of self-tuning indexes using reinforcement learni (RL).**
   - **Key Innovation: UpLIF and LITune for autonomous index optimization.**
   - **Future Direction: Continuous online learning for real-time adaptability**





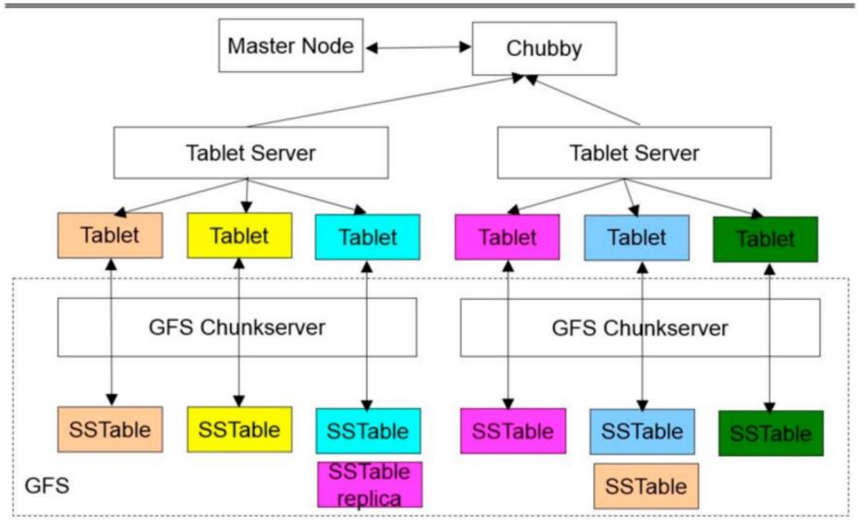


[5] Heidari, A., Lissandrini, M., & Aref, W. G. (2024). UpLIF: An Updatable Self-Tuning Learned Index Framework. arXiv preprint arXiv:2408.04113. https://arxiv.org/abs/2408.04113

Cloud-Native Adoption and Scalability Challenges

1. **Learned Indexes in Cloud-Native and Distributed Databases**
   - **Trend:** Increasing adoption in **cloud-native databases** for efficiency.
   - **Key Innovation: Learned indexing layers** in Google Bigtable for lower latency.
   - **Future Direction: Self-driving indexing services** for cloud environments.
2. **Reducing Retraining Costs and Enhancing Scalability**
   - **Trend:** Minimizing **retraining overhead** and memory use in mutable indexes.
   - **Key Innovation: Amortized retraining** and partial updates.
   - **Future Direction: Distributed learned indexes** for scalable cloud deployments.

BigTable Architecture



[6] Wang, W., Zhang, Y., Chen, S., & Li, C. (2022). Are Updatable Learned Indexes Ready?. Proceedings of the VLDB Endowment, 15(11), 3004-3016. https://doi.org/10.14778/3551793.3551811

References

[1] Ding, J., Minhas, U. F., Yu, J., Wang, C., Do, J., Li, Y., Zhang, H., Chandramouli, B., Gehrke, J., Kossmann, D., Lomet, D., & Kraska, T. (2020). ALEX: An Updatable Adaptive Learned Index. In Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (SIGMOD'20), June 14–19, 2020, Portland, OR, USA. ACM, New York, NY, USA. https://doi.org/10.1145/3318464.3389711

[2] Kraska, T., Beutel, A., Chi, E. H., Dean, J., & Polyzotis, N. (2018). The Case for Learned Index Structures. *arXiv:1712.01208 [cs.DB]*.

[3] Mamun, A. A., Wu, H., & Aref, W. G. (n.d.). A tutorial on learned multi-dimensional indexes. https://www.cs.purdue.edu/homes/aref/LMDI2020/LMDI_Tutorial_SIGSpatial2020.pdf

[4] H. Abu-Libdeh, D. Altınbüken, A. Beutel, E. H. Chi, L. Doshi, T. Kraska, X. (S.) Li, A. Ly, and C. Olston, "Learned Indexes for a Google-scale Disk-based Database," Workshop on ML for Systems, Vancouver, Canada, December 2020.

[5] Heidari, A., Lissandrini, M., & Aref, W. G. (2024). UpLIF: An Updatable Self-Tuning Learned Index Framework. arXiv preprint arXiv:2408.04113. https://arxiv.org/abs/2408.04113

[6] Wang, W., Zhang, Y., Chen, S., & Li, C. (2022). Are Updatable Learned Indexes Ready?. Proceedings of the VLDB Endowment, 15(11), 3004-3016. https://doi.org/10.14778/3551793.3551811

[7] Nathan, V., Ding, J., Alizadeh, M., & Kraska, T. (2020). Learning multi-dimensional indexes. Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, 985–1000. https://doi.org/10.1145/3318464.3380579