How does ML benefit traditional data structure

Zhaoguo Wang

The Case for Learned Index Structures. SIGMOD'18 GOOGLE
XIndex: A Scalable Learned Index for Multicore Data Storage. PPoPP'20. STJU-IPADS

Questions to Answer

1. How to index data with ML models. } Learned Index (SIGMOD'18)
2. How does it benefit the performance? } XIndex (PPoPP'20)
3. What are the limitations?
4. How to break the limitations?

Questions to Answer

1. How to index data with ML models.

Learned Index (SIGMOD'18)

2. How does it benefit the performance?

3. What are the limitations?

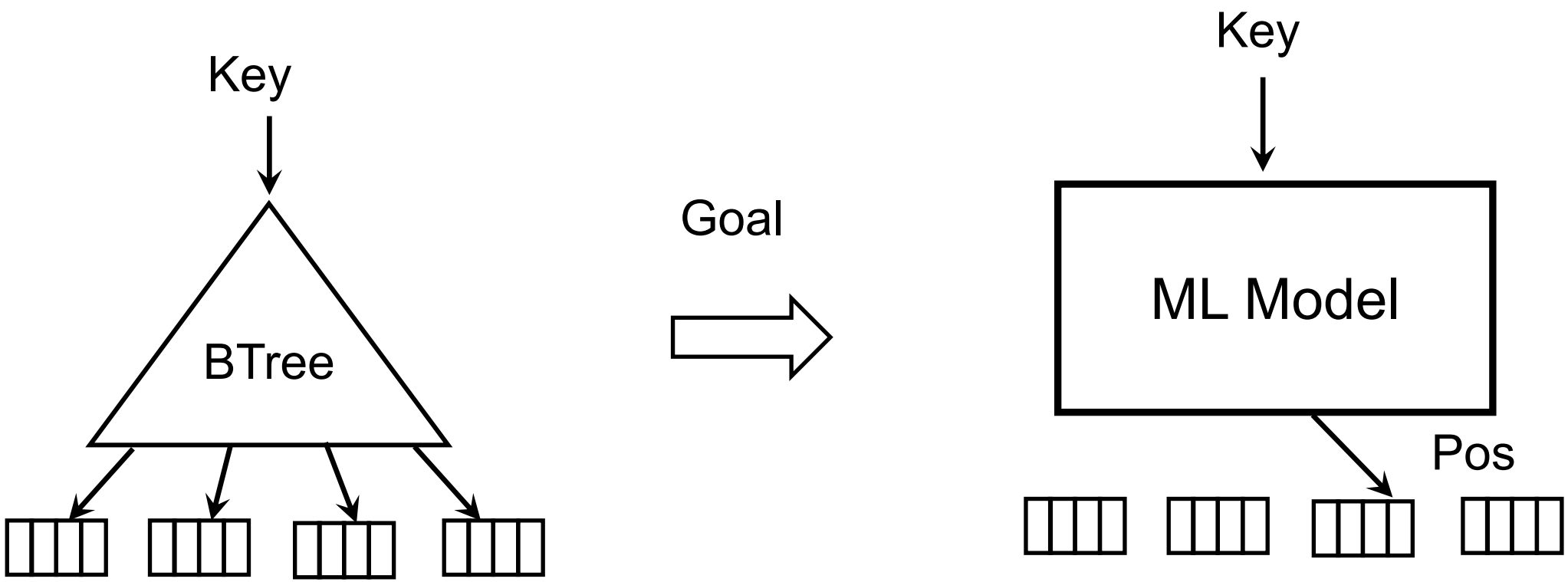
4. How to break the limitations?

XIndex (PPoPP'20)

Goal – Replace B+ Tree With Learning Model

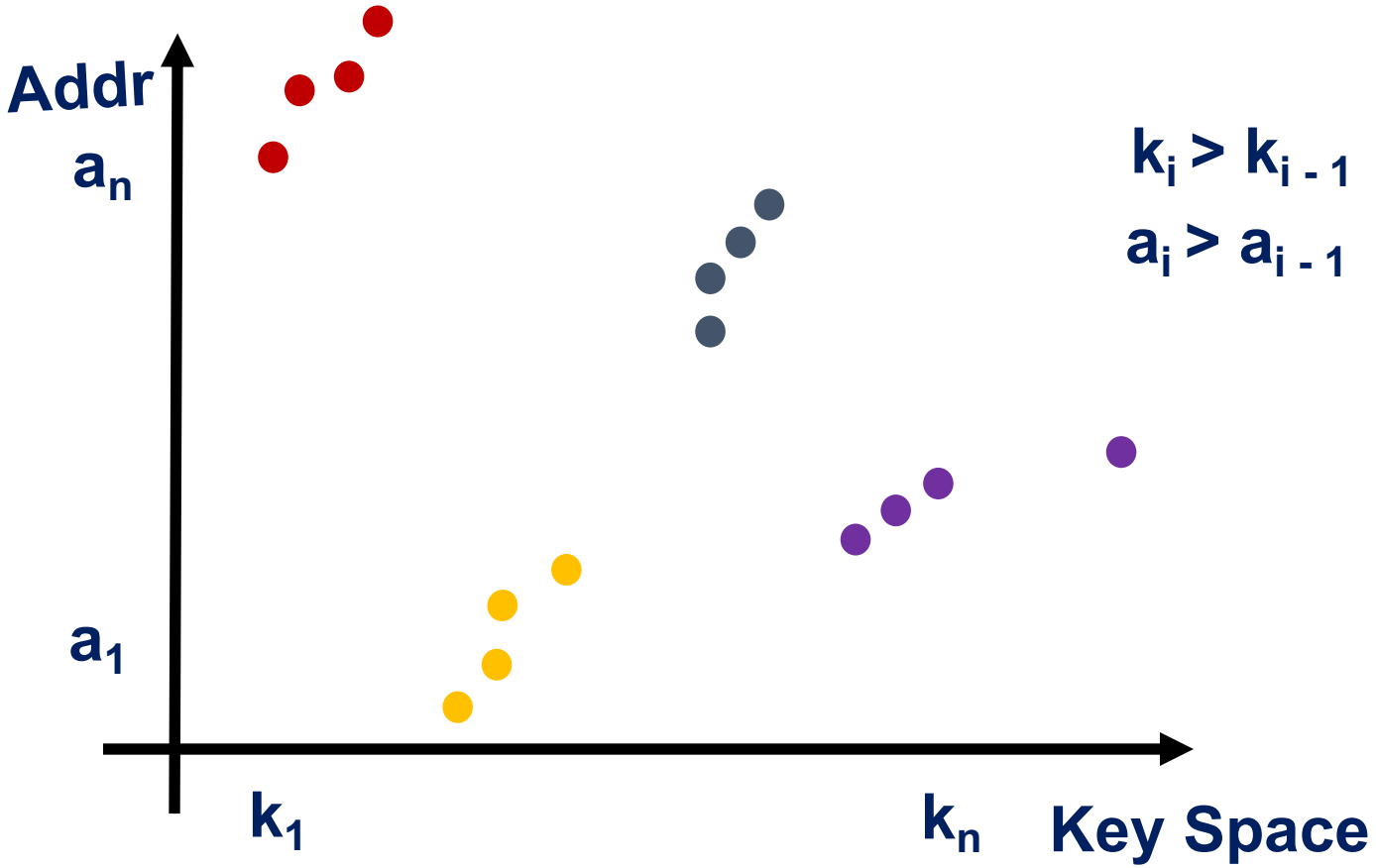
What is the intuition?

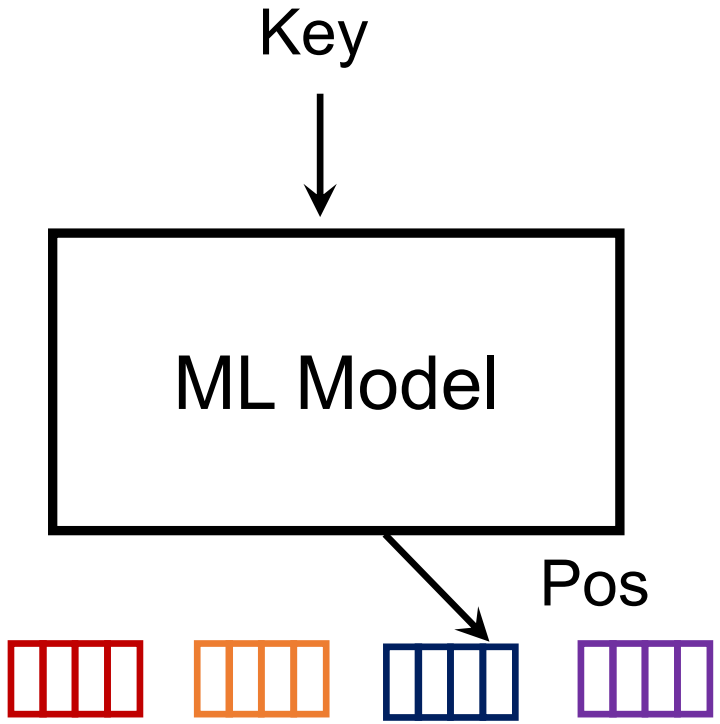
Index can be considered as `addr = f(key)`, which we should learn.



Goal – Replace B+ Tree With Learning Model

Issue: The distribution looks to be complicated, we need to use complex model to simulate the distribution.

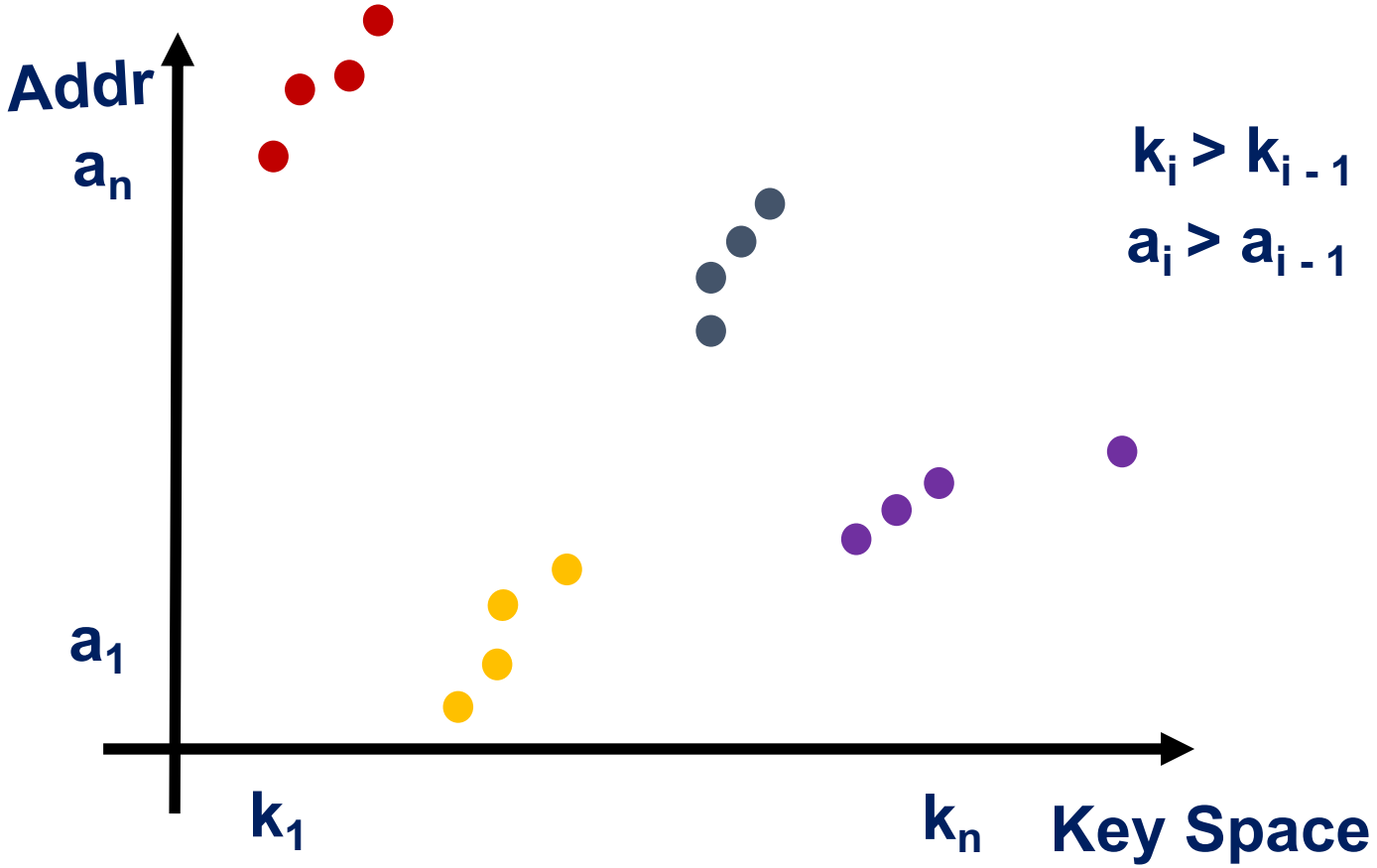


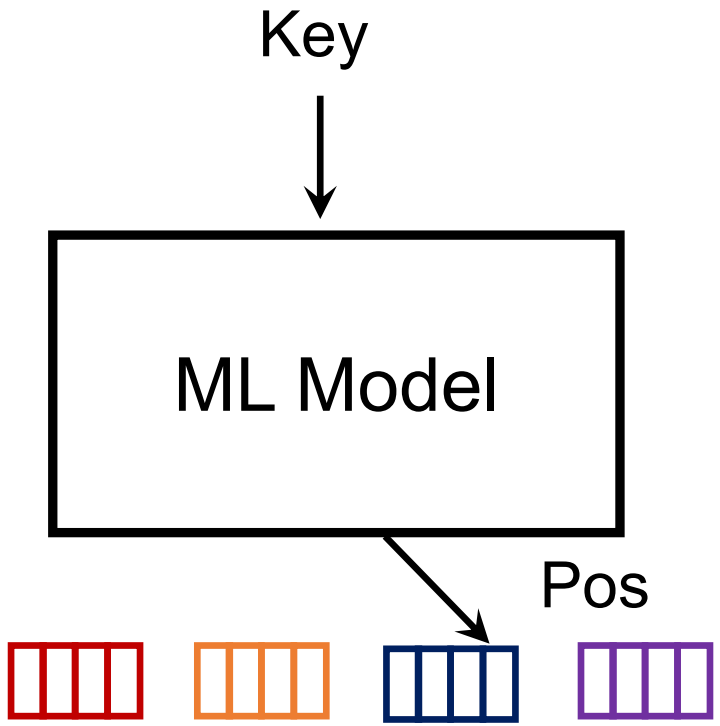


Goal – Replace B+ Tree With Learning Model

Question: Can we simplify the distribution?

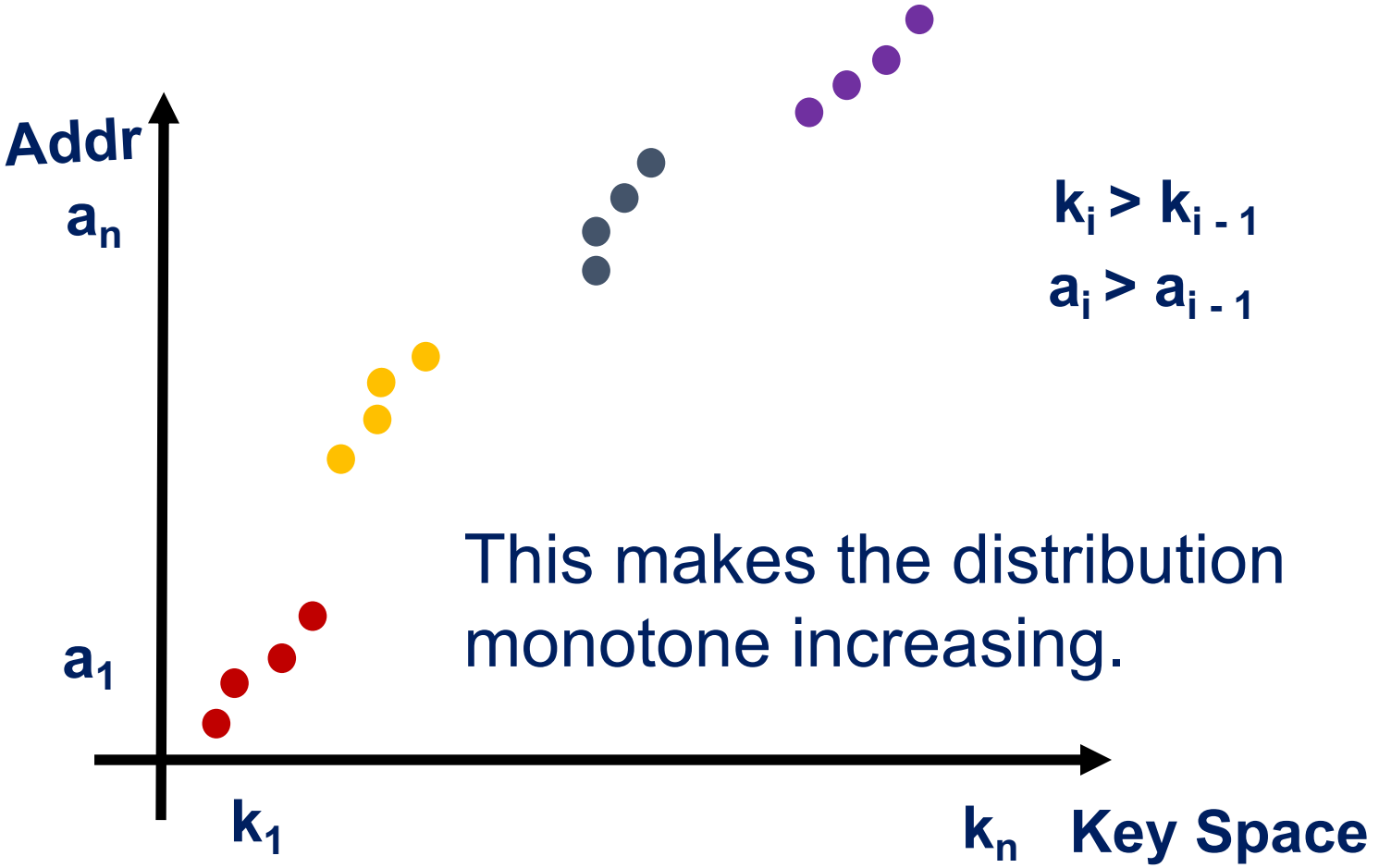
Answer: We do the simplification by adding restrictions on the data.

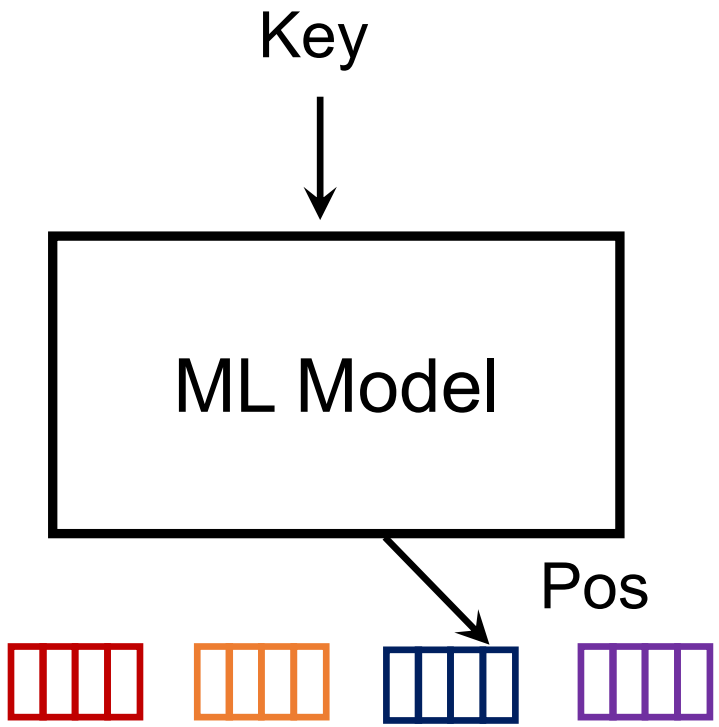




Simplify The Distribution

Add restriction: k_i's addr must be larger than k_{i-1}'s addr.



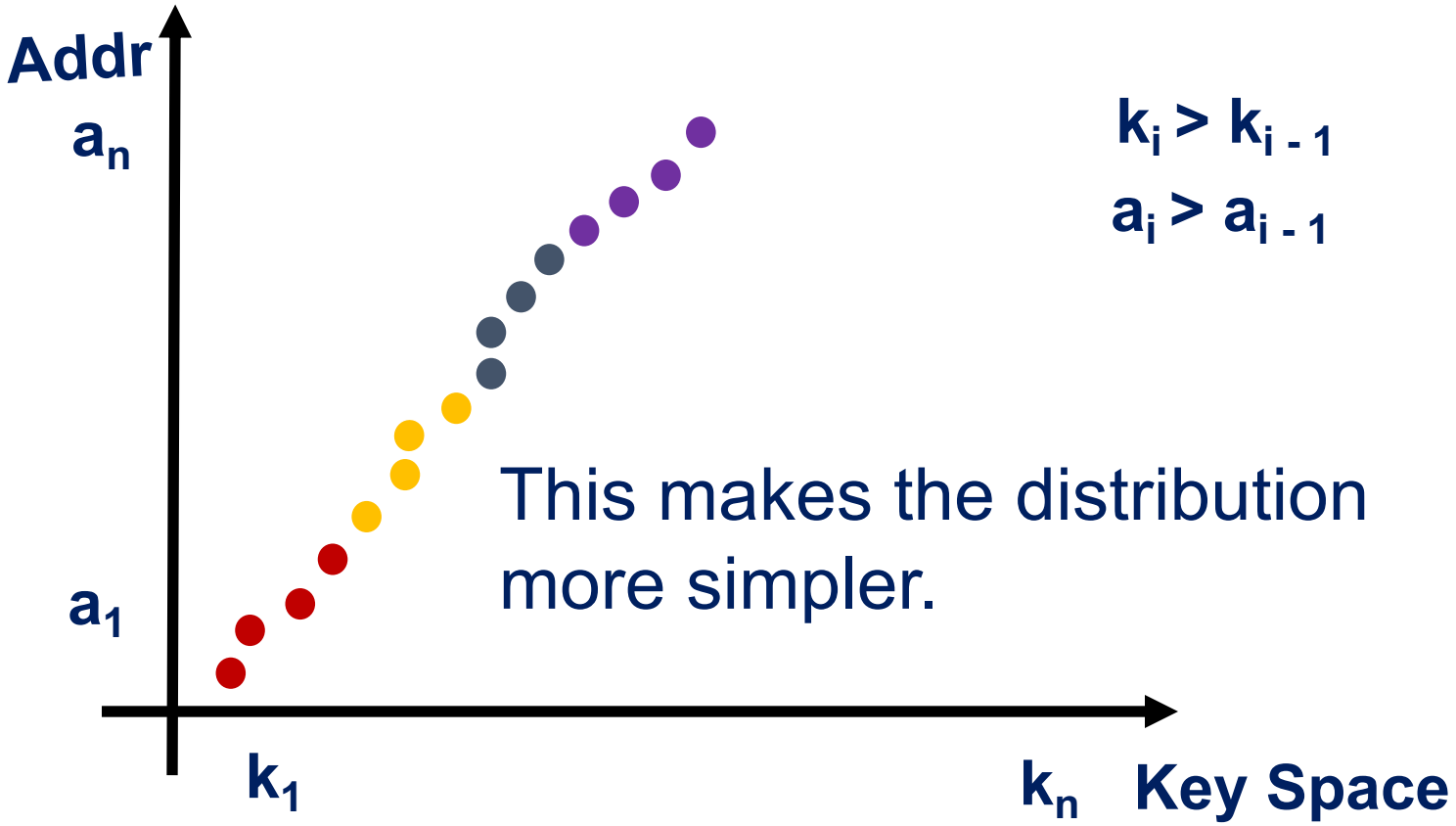


Simplify The Distribution

Add restriction: k_i's addr must be larger than k_{i-1}'s addr.

More restriction: the entire memory should be contiguous

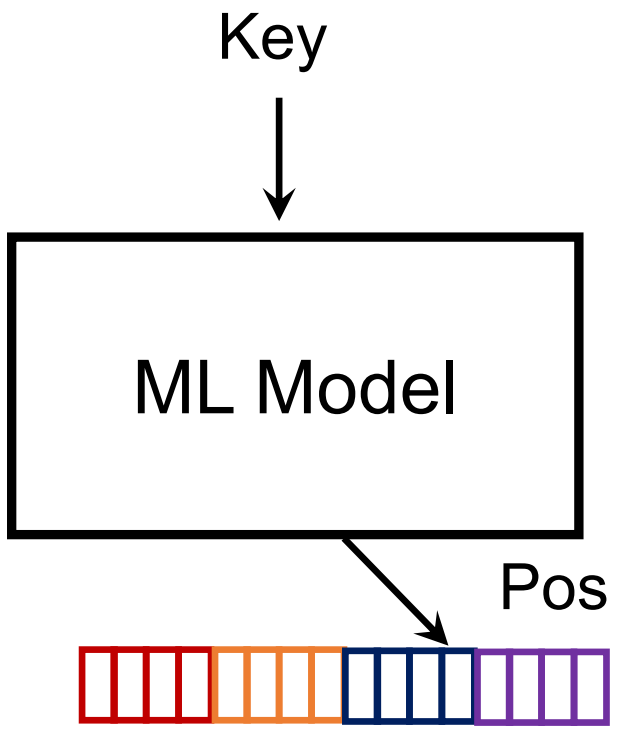
Data should be contiguous.



$k_i > k_{i-1}$

$a_i > a_{i-1}$

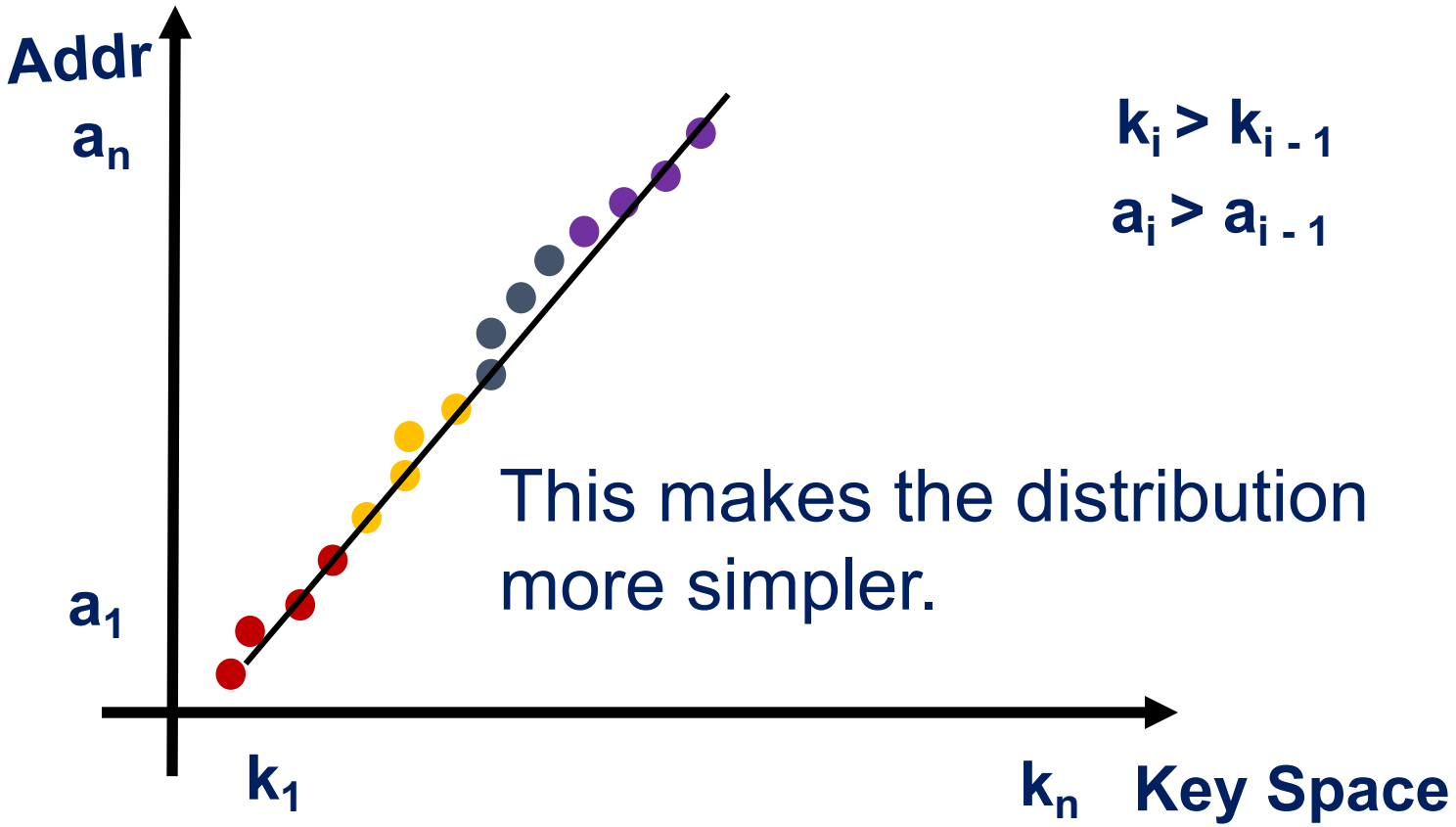
This makes the distribution more simpler.



Using Linear Model To Simulate The Distribution

We can use a linear model to learn the distribution.

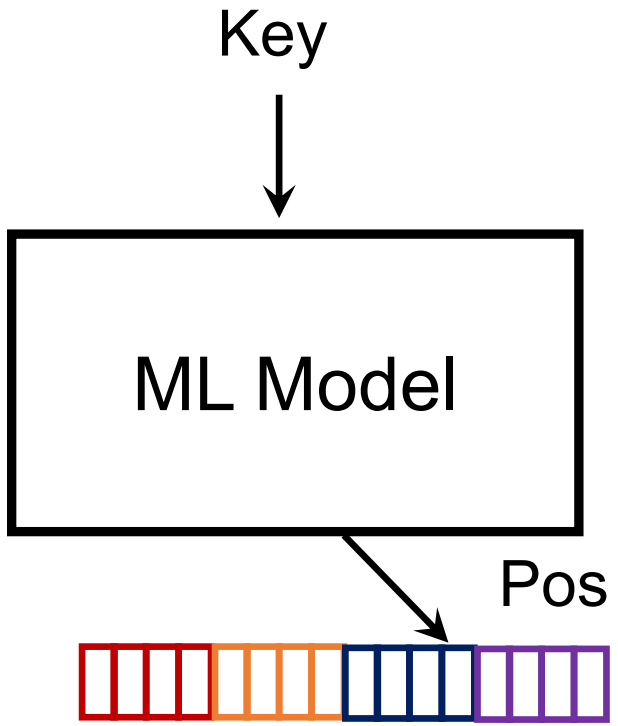
$pos_{predict} = w * key + b$, Loss = $\frac{1}{N}\sum_{i=0}^{n}(pos_{predict} - pos_{real})^2$



$k_i > k_{i-1}$

$a_i > a_{i-1}$

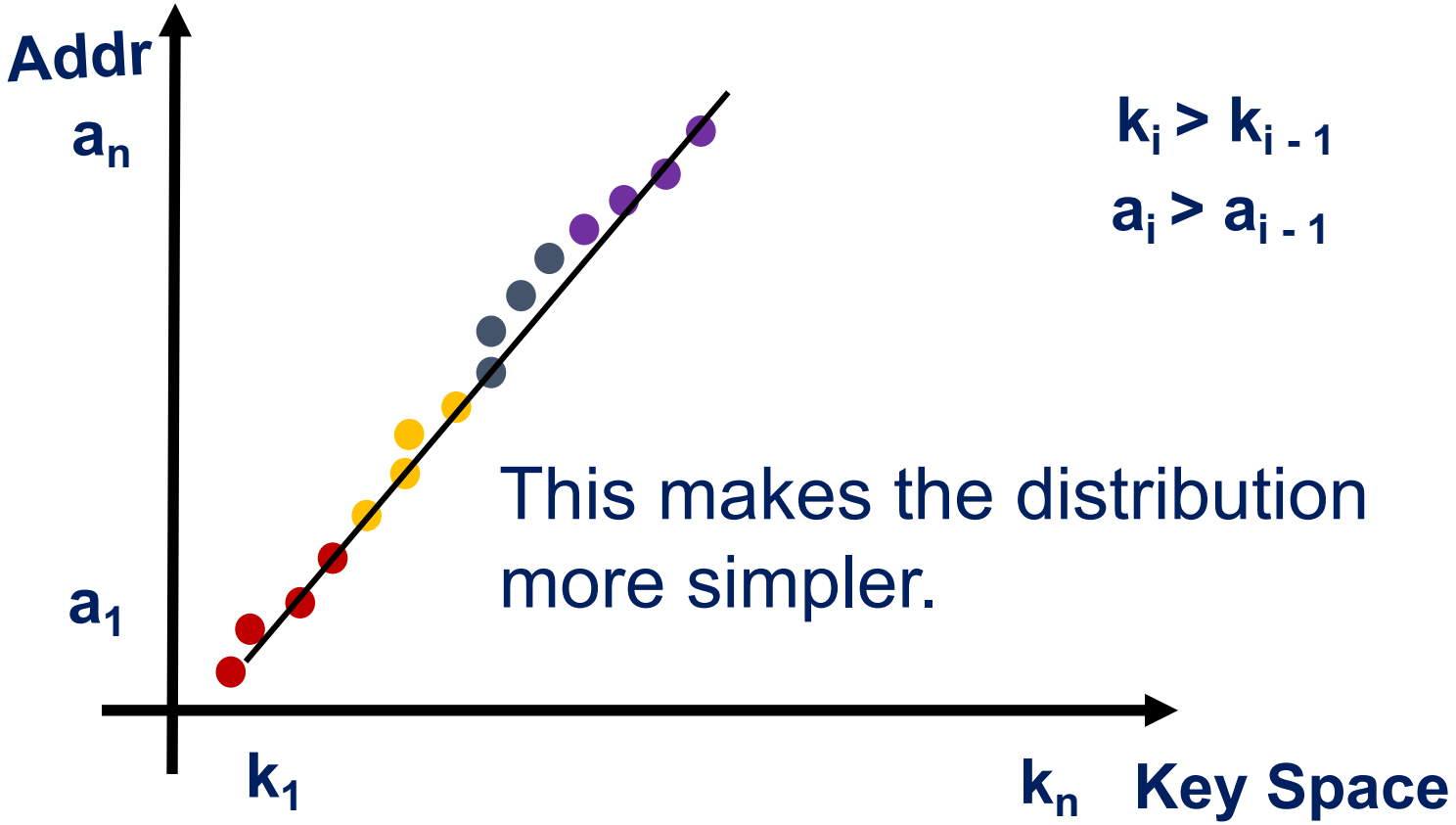
This makes the distribution more simpler.



Correct The Predict Result

Problem: the predicted position is different with the real position.

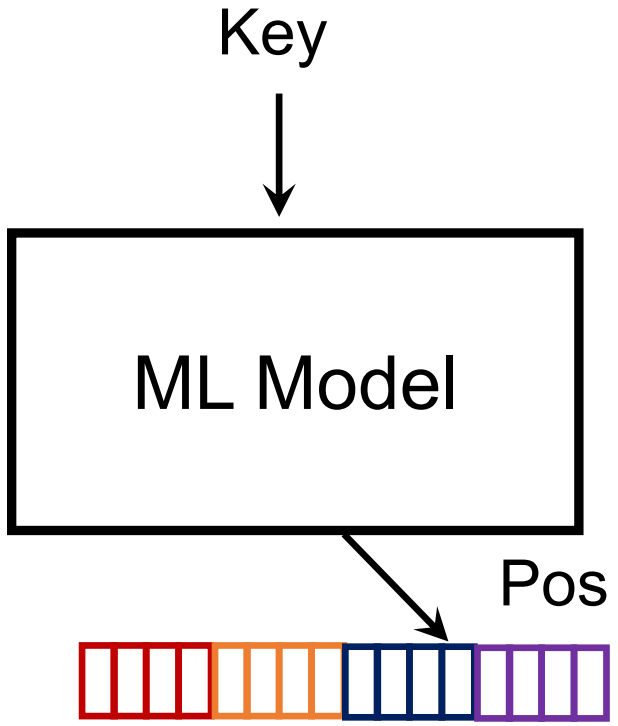
Solution: perform the binary search on misprediction.



$k_i > k_{i-1}$

$a_i > a_{i-1}$

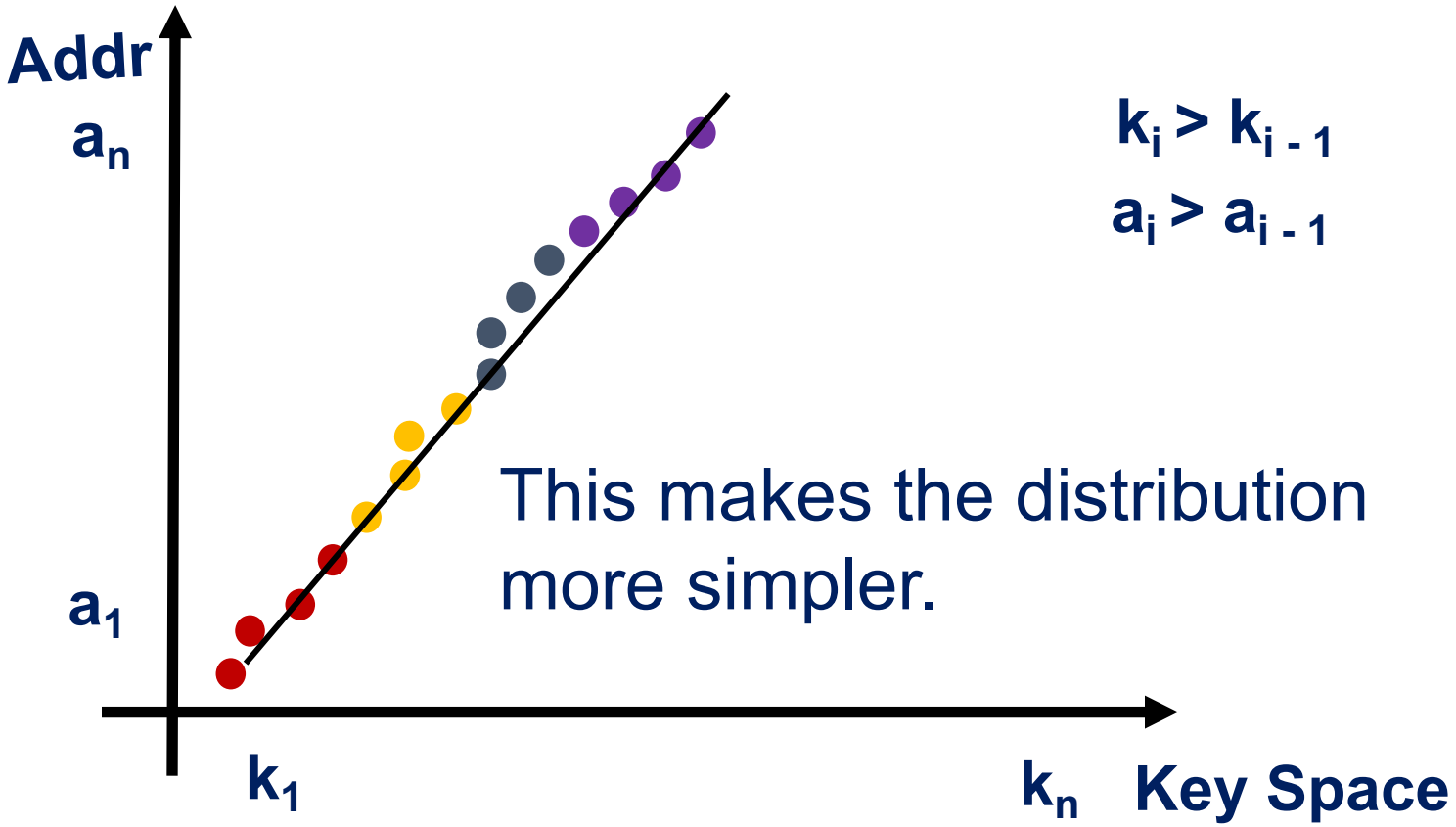
This makes the distribution more simpler.



Correct The Predict Result

Problem: the predicted position is different with the real position.

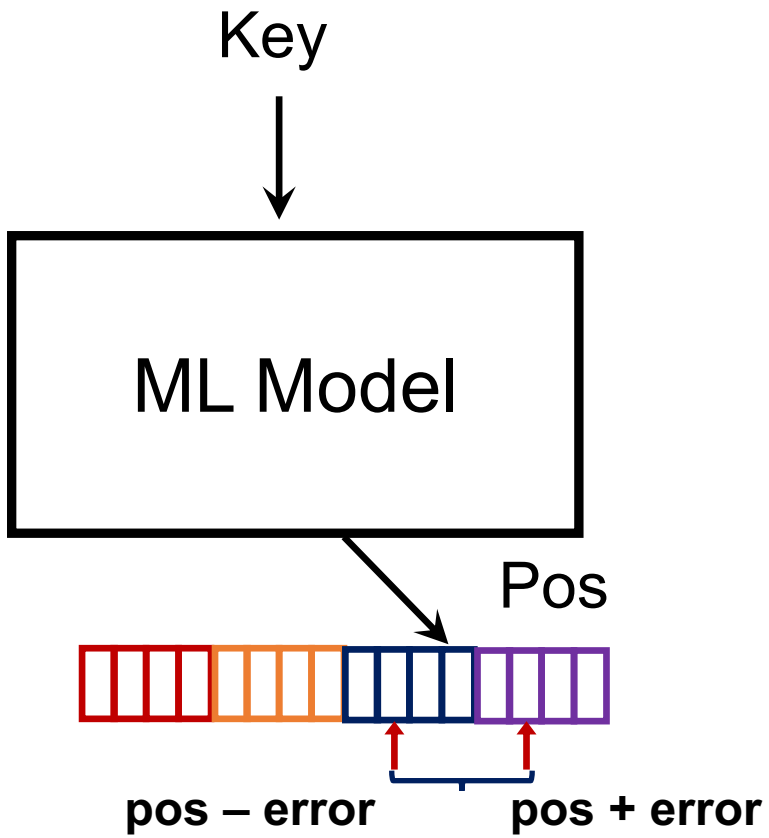
Solution: perform the binary search in a fixed range.



$k_i > k_{i-1}$

$a_i > a_{i-1}$

This makes the distribution more simpler.



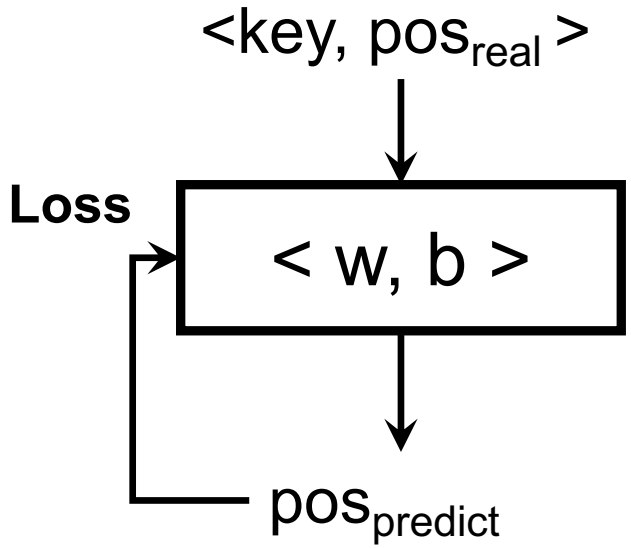
Correct The Predict Result

Problem: How to know the error for keys not in the training set?

Solution: All keys must be known beforehand (Restriction II).

Phase I: Train the model.

- Use *L* to correct the model
- Or directly calculate the w, b



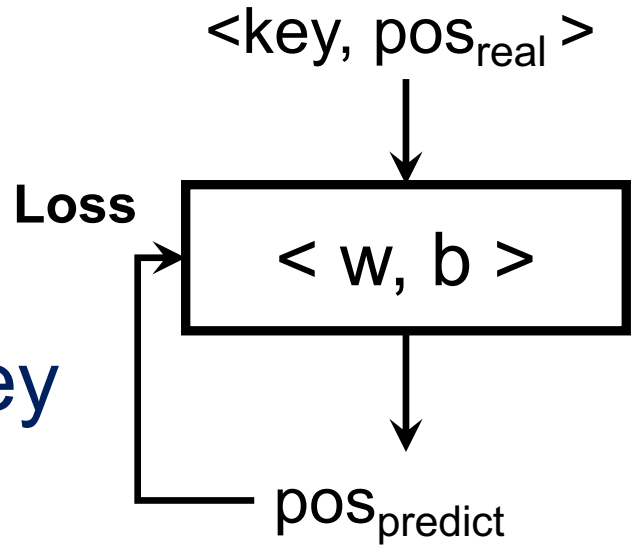
Correct The Predict Result

Problem: How to know the error for keys not in the training set?

Solution: All keys must be known beforehand (Restriction II).

Phase I: Train the model.

- Use *L* to correct the model
- Or directly calculate the w, b



Phase II: Calculate the error bound for each key

- Should include all keys in the workload
- Error = | $pos_{predict}$ – pos_{real} |

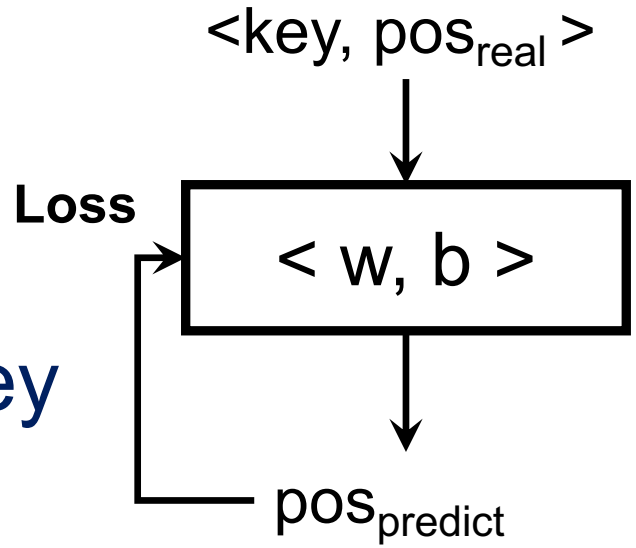
Correct The Predict Result

Problem: How to know the error for keys not in the training set?

Solution: All keys must be known beforehand (Restriction II).

Phase I: Train the model.

- Use *L* to correct the model
- Or directly calculate the w, b



Phase II: Calculate the error bound for each key

- Should include all keys in the workload
- Error = | $pos_{predict}$ – pos_{real} |

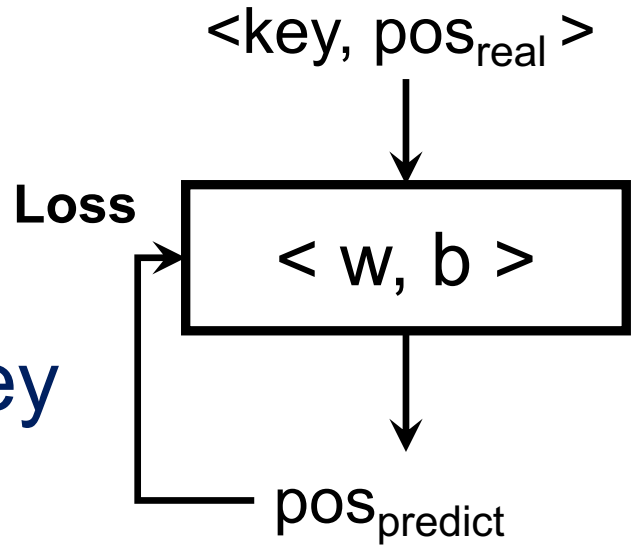
Correct The Predict Result

Problem: How to know the error for keys not in the training set?

Solution: All keys must be known beforehand (Restriction II).

Phase I: Train the model.

- Use *L* to correct the model
- Or directly calculate the w, b



Phase II: Calculate the error bound for each key

- Should include all keys in the workload
- Error = | $pos_{predict}$ – pos_{real} |

Issue: It takes cost to maintain an error for every key.

Solution: we only need to keep the max error among all records (keys).

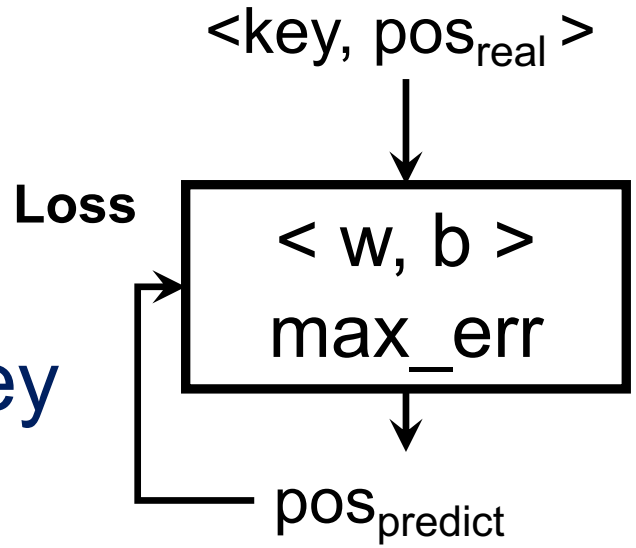
Correct The Predict Result

Problem: How to know the error for keys not in the training set?

Solution: All keys must be known beforehand (Restriction II).

Phase I: Train the model.

- Use *L* to correct the model
- Or directly calculate the w, b



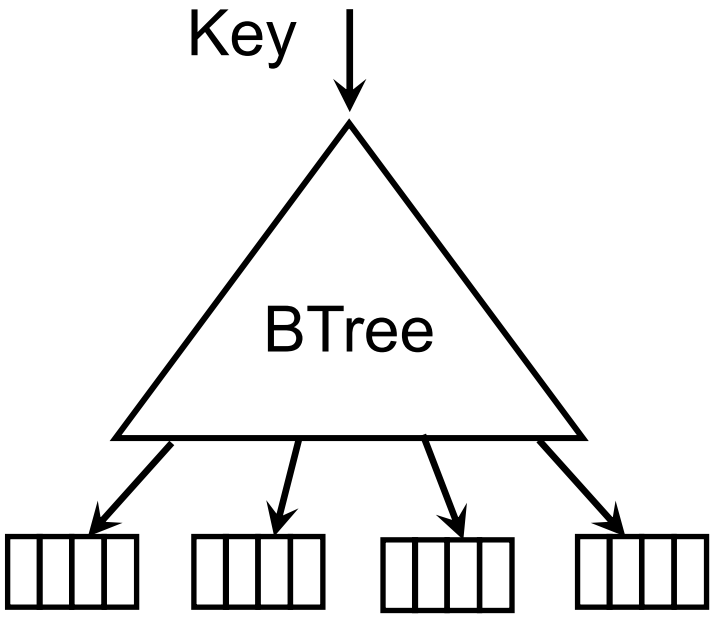
Phase II: Calculate the error bound for each key

- Should include all keys in the workload
- Error = | $pos_{predict} - pos_{real}$ |

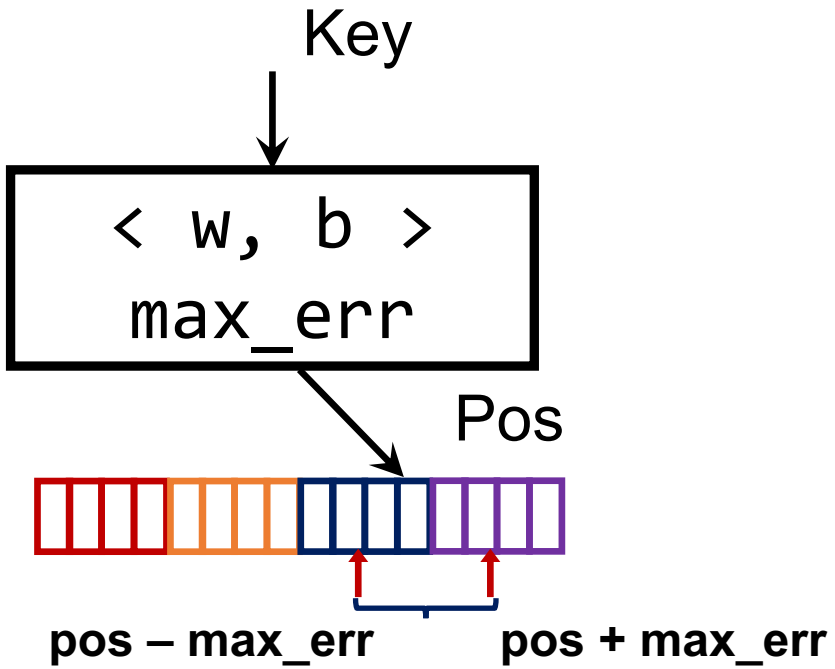
Issue: It takes cost to maintain an error for every key.

Solution: we only need to keep the max error among all records (keys).

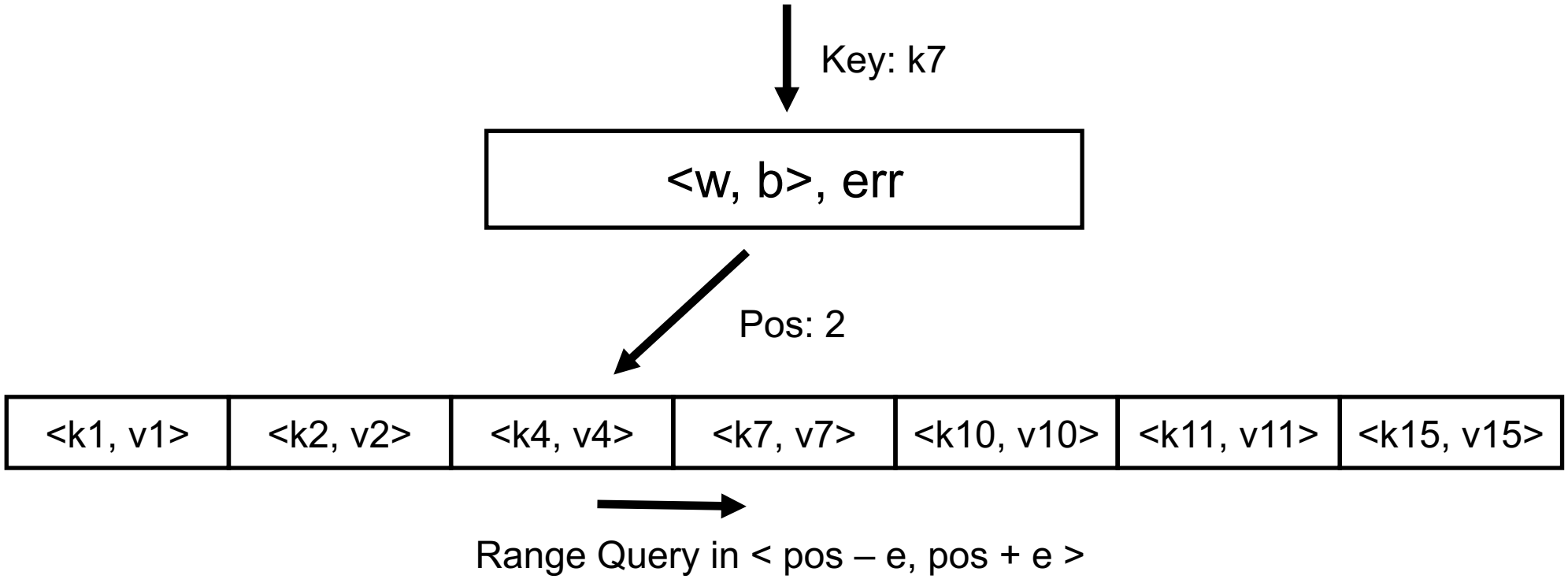
Correct The Predict Result



Adding restrictions to simplify the distribution.



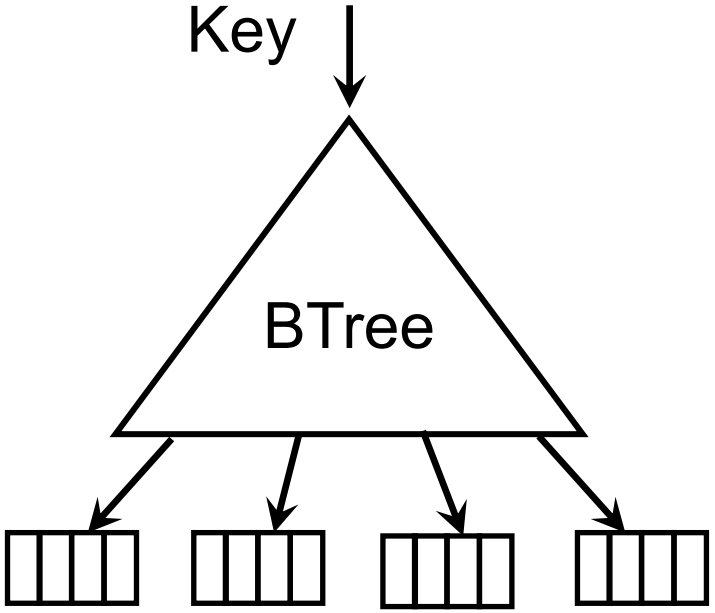
Example


Key: k7
<w, b>, err
Pos: 2
<k1, v1>
<k2, v2>
<k4, v4>
<k7, v7>
<k10, v10>
<k11, v11>
<k15, v15>
Range Query in < pos – e, pos + e >

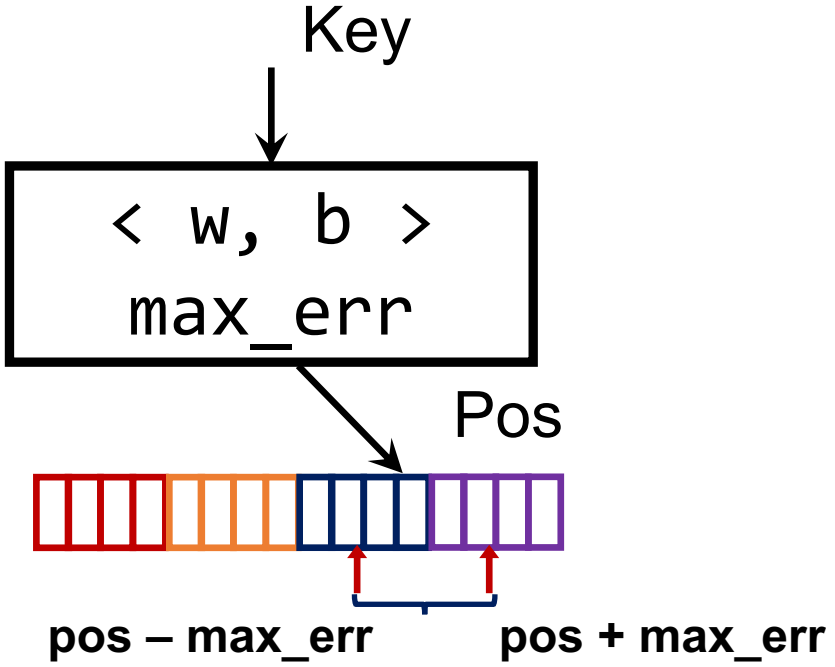
Correct The Predict Result

Problem: The error bound is still large.

Solution: Leverage multi-stage design.



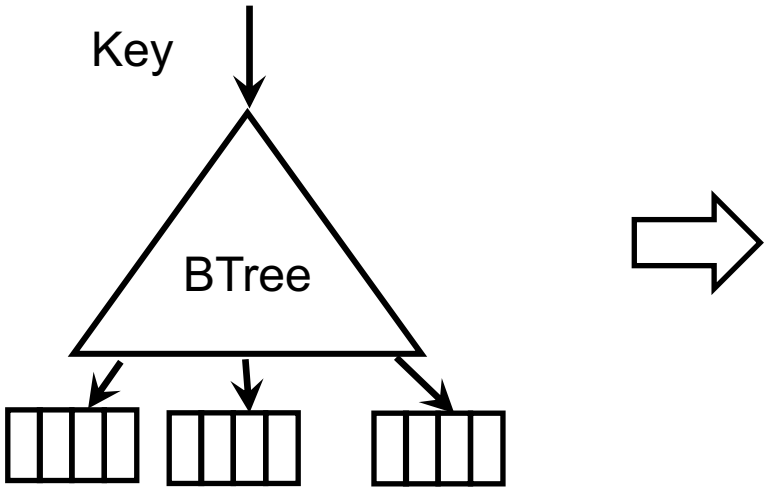
Adding restrictions to simplify the distribution.

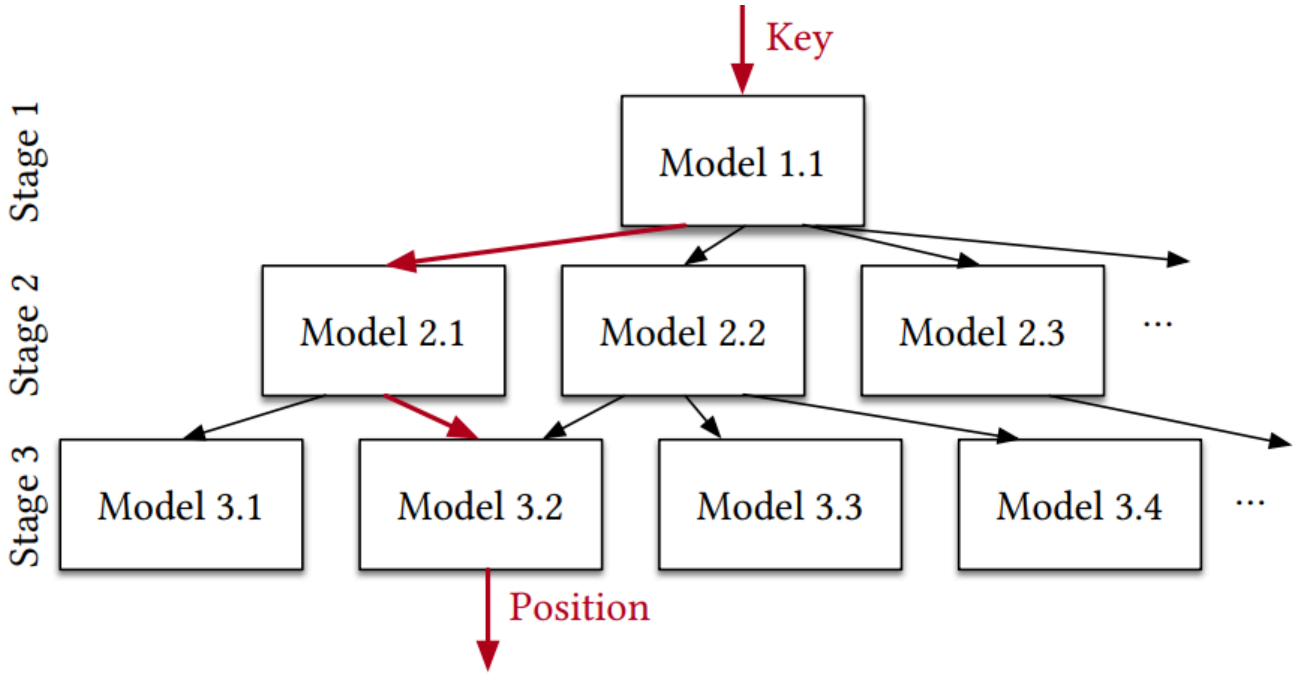


Multi-stage Design

Problem: The error bound is still large.

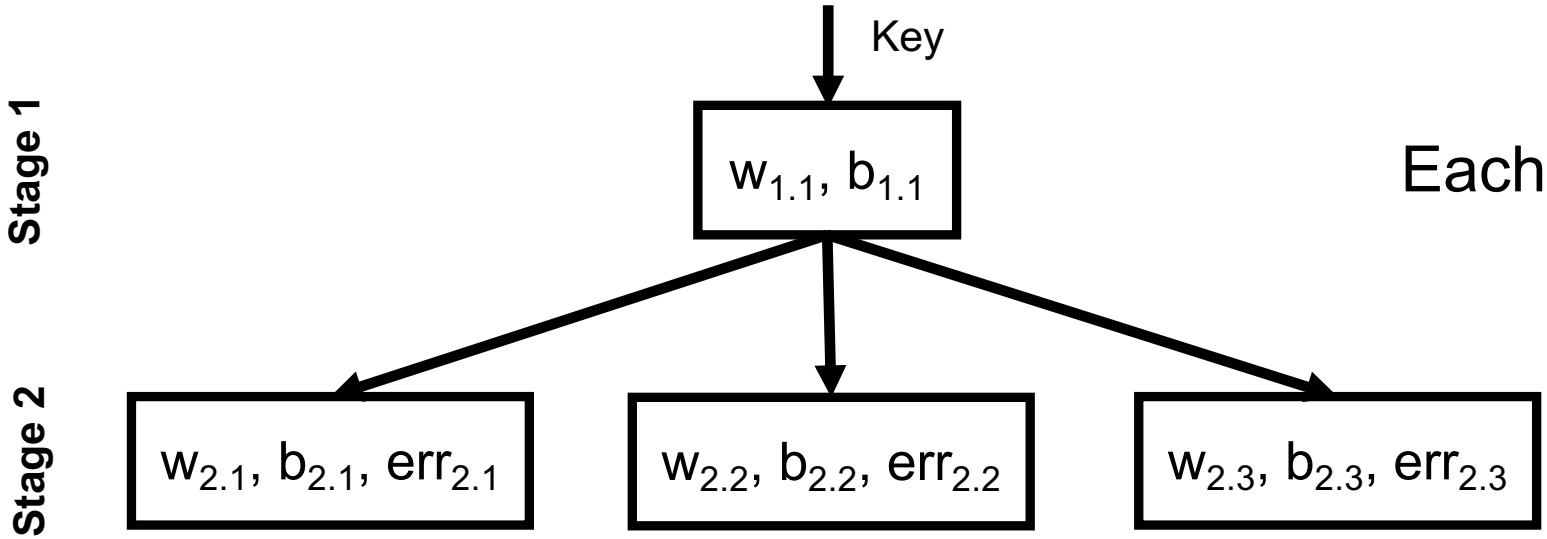
Solution: Leverage multi-stage design.





Every model `i` has its own parameter: $\langle w_i, b_i \rangle$, err_i

Multi-stage Design



Each model can be trained individually.

Inference Process:

Step 1. Predict the position with $model_{1.1}$ ($pos_1 = key * w_{1.1} + b_{1.1}$).

Step 2. Use pos_1 to pickup a model in stage 2 (m = pos_1 / N * M_2 , N is the total number of records,

M_i is the total number of models at stage 2)

Step 3. Predict the position with the mode x in stage 2 ($pos_2 = key * w_{2.x} + b_{2.x}$).

Step 4. Correct the position with the $err_{2.x}$

Performance of Learned Index

		Map Data			Web Data			Log-Normal Data		
Type	**Config**	Size (MB)	Lookup (ns)	Model (ns)	Size (MB)	Lookup (ns)	Model (ns)	Size (MB)	Lookup (ns)	Model (ns)
Btree	page size: 32	52.45 (4.00x)	274 (0.97x)	198 (72.3%)	51.93 (4.00x)	276 (0.94x)	201 (72.7%)	49.83 (4.00x)	274 (0.96x)	198 (72.1%)
	page size: 64	26.23 (2.00x)	277 (0.96x)	172 (62.0%)	25.97 (2.00x)	274 (0.95x)	171 (62.4%)	24.92 (2.00x)	274 (0.96x)	169 (61.7%)
	page size: 128	13.11 (1.00x)	265 (1.00x)	134 (50.8%)	12.98 (1.00x)	260 (1.00x)	132 (50.8%)	12.46 (1.00x)	263 (1.00x)	131 (50.0%)
	page size: 256	6.56 (0.50x)	267 (0.99x)	114 (42.7%)	6.49 (0.50x)	266 (0.98x)	114 (42.9%)	6.23 (0.50x)	271 (0.97x)	117 (43.2%)
	page size: 512	3.28 (0.25x)	286 (0.93x)	101 (35.3%)	3.25 (0.25x)	291 (0.89x)	100 (34.3%)	3.11 (0.25x)	293 (0.90x)	101 (34.5%)
Learned Index	2nd stage models: 10k	0.15 (0.01x)	98 (2.70x)	31 (31.6%)	0.15 (0.01x)	222 (1.17x)	29 (13.1%)	0.15 (0.01x)	178 (1.47x)	26 (14.6%)
	2nd stage models: 50k	0.76 (0.06x)	85 (3.11x)	39 (45.9%)	0.76 (0.06x)	162 (1.60x)	36 (22.2%)	0.76 (0.06x)	162 (1.62x)	35 (21.6%)
	2nd stage models: 100k	1.53 (0.12x)	82 (3.21x)	41 (50.2%)	1.53 (0.12x)	144 (1.81x)	39 (26.9%)	1.53 (0.12x)	152 (1.73x)	36 (23.7%)
	2nd stage models: 200k	3.05 (0.23x)	86 (3.08x)	50 (58.1%)	3.05 (0.24x)	126 (2.07x)	41 (32.5%)	3.05 (0.24x)	146 (1.79x)	40 (27.6%)

Learned Index has better performance and fewer memory cost.

Questions to Answer

1. How to index data with ML models. } Learned Index (SIGMOD'18)

2. How does it benefit the performance?

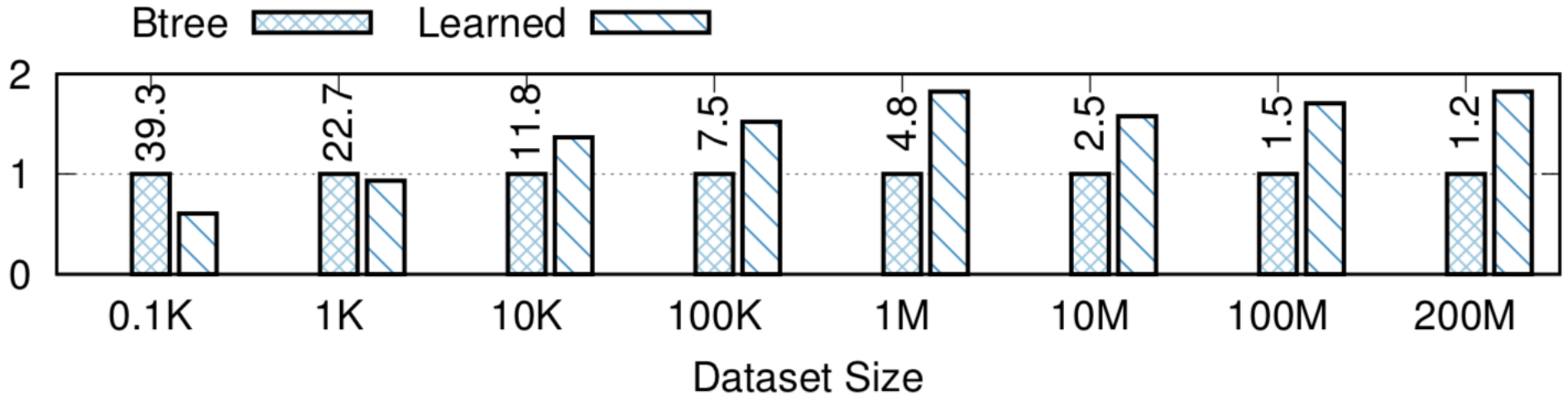
3. What are the limitations?

4. How to break the limitations?

} XIndex (PPoPP'20)

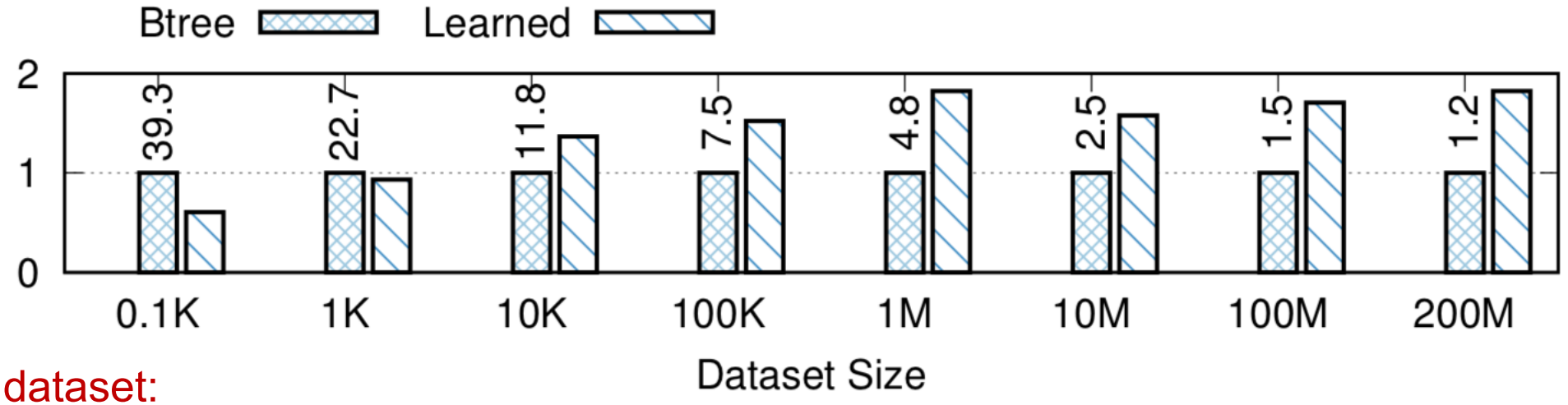
Why Benefit The Performance?

Learned index has better performance for large dataset.



Why Benefit The Performance?

Learned index has better performance for large dataset.



Small dataset:

For single query, learned index executes more instructions with higher CPI comparing with B-Tree.

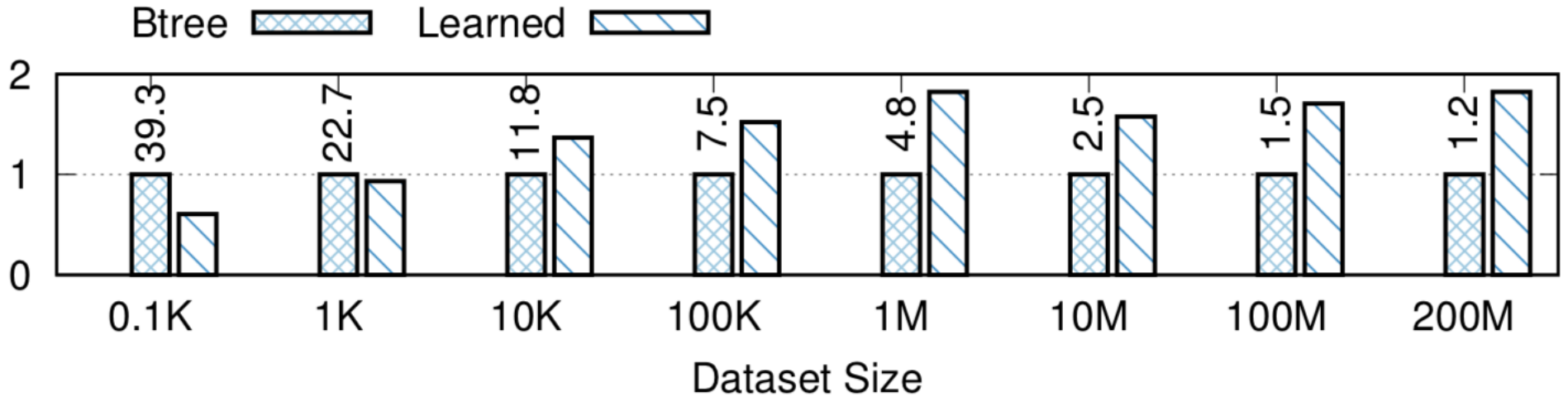
Example: 0.1K dataset,

#instruction / Query: 142 v.s. 94

CPI: 0.4 v.s. 0.29

Why Benefit The Performance?

Learned index has better performance for large dataset.



Large dataset:

For single query, Learned index executes less instructions with better cache locality.

Reason:

1. BTree's #instructions increased is faster than Learned Index.
   Btree - 33 (14) inst. by adding a layer with fanout of 16 (2).
   Learned Index – 4 inst. when the error is increased by 1.

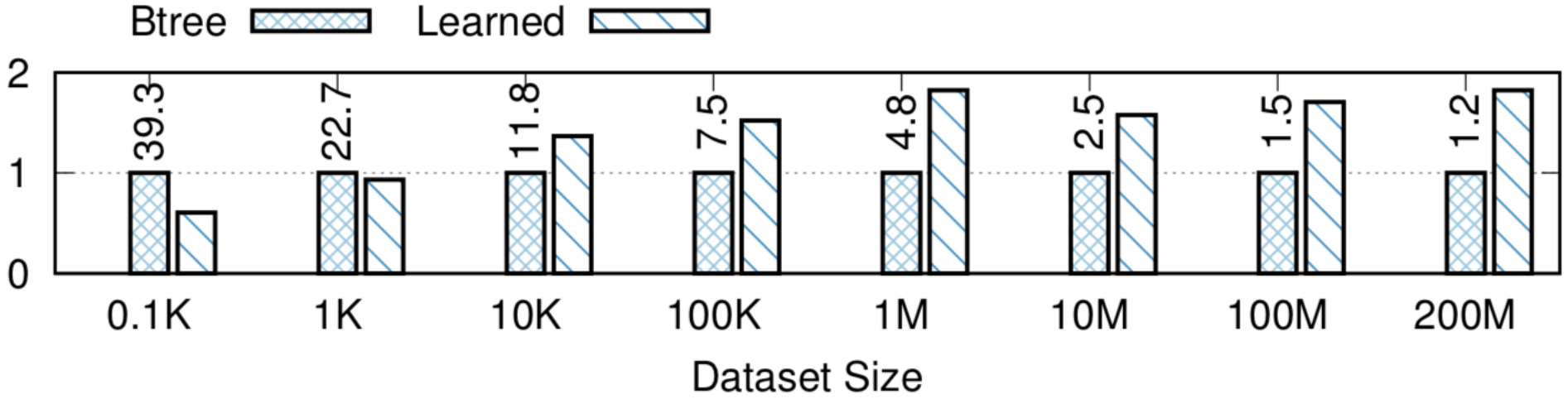
Example: 1K → 1M

#Layers in Btree: 3 → 6

Error bound in Learned Index: 0.6 → 4.76

Why Benefit The Performance?

Learned index has better performance for large dataset.



Large dataset:

For single query, Learned index executes less instructions with better cache locality.

Reason:

2. Learned index has better cache locality because of its small size.

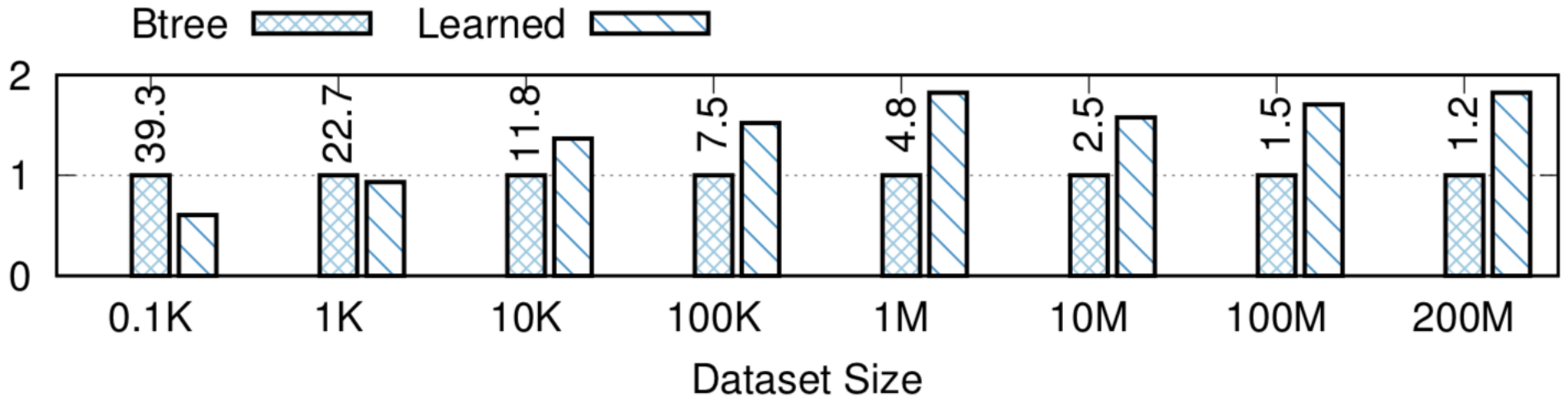
Example: 200MB

Btree: 37 cache lines per query

Learned Index: 10 cache lines per query

Why Benefit The Performance?

Learned index has better performance for large dataset.



Small dataset:
For single query, learned index executes more instructions with higher CPI comparing with B-Tree.

Large dataset:
For single query, Learned index executes less instructions with better cache locality.

Goodbye & Good Luck