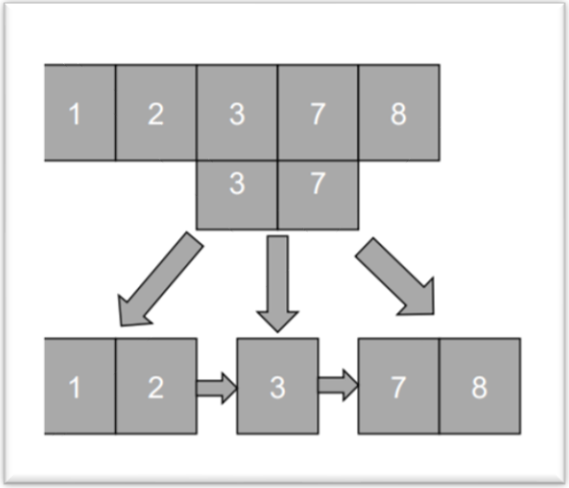
From WiscKey to Bourbon: A Learned Index for Log-Structured Merge Trees

OSDI20

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Indexes

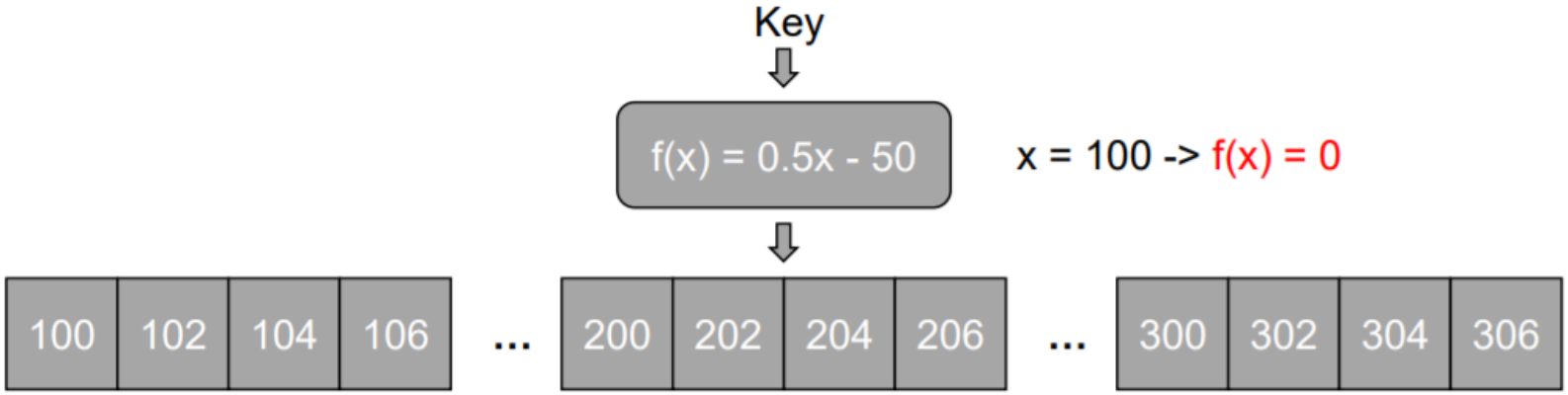
- 在计算机系统中，查询数据是一个很重要的部分
- 传统的方法，使用特定的数据结构来便于查询
  - 在一个B+ tree中查询一个数据，时间复杂度在O(logN)



- 一个问题:假设我们提前知道数据会怎样?

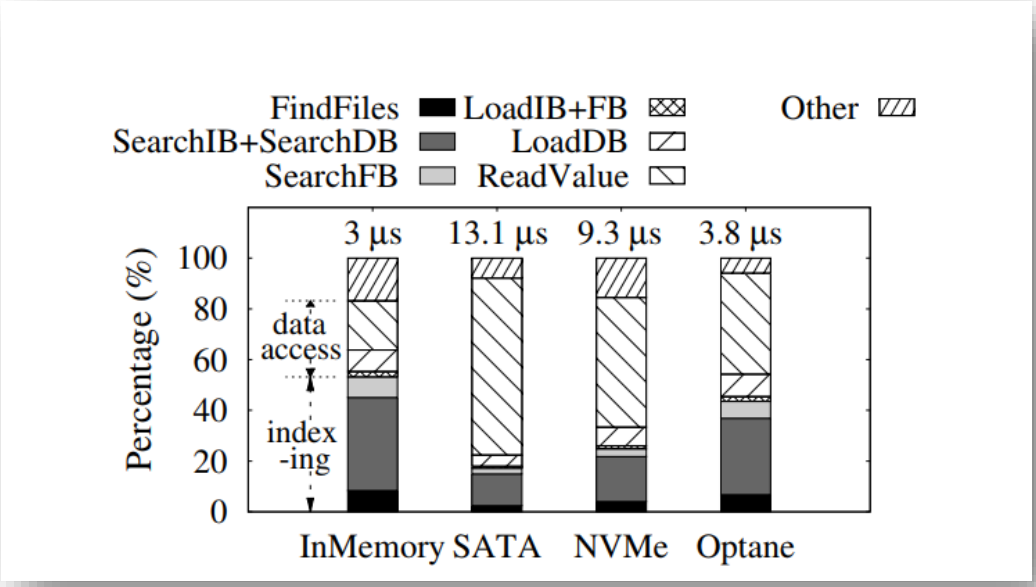
Bring Learning to Indexing

- 如果我们知道数据分布，查询数据可以更快
  - 用机器学习模型学的数据分布
  - 查询时间复杂度在O(1)
  - 查询空间复杂度在O(1)



Learned Indexes for Look-up

- Learned index可以提高look-up性能



Learned Indexes for Writes

- Learned indexes可以提高查询性能，但是之前的关于learned indexes不支持数据更新
  - 因为数据更新会打乱数据分布，模型必须重新训练，从而影响性能
- 一个思考:LSMs会涉及到数据更新，learned indexes是否适用于LSMs?
- 一些观察:
  - 数据更新回改变LSM tree一部分的数据位置，但是大部分数据保持不变
  - 不变的sstable file适合learned indexes
  - sstable文件是有序的，只需要简单的模型训练就行

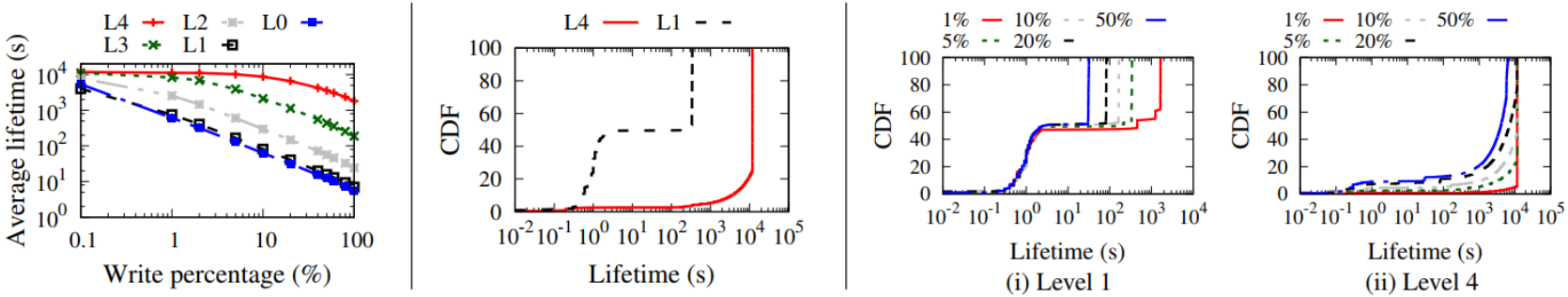
Learning Guidelines

- 在sstable文件的学习粒度(learning granularity)
  - 不需要更新模型
  - 模型有固定的精度

- 在学习之前考虑的因素
  - SSTable的寿命
    - 一个模型能使用多久
  - 在SStable的查询次数
    - 一个模型的有效频率是多少

Learning Guidelines

- SSTable的生命周期
  - L0:平均10秒
  - L4:平均1小时
  - 一些短暂的表: 小于1秒



(a) Average lifetimes with varying write % (b) Lifetime distribution with 5% writes

(c) Lifetime distributions with varying write %

Learning Guidelines

- 指导原则1:偏爱低level的表
  - 处于更低level的文件生存时间更长
- 指导原则2:在学习之前先等待
  - 避免学习到寿命很短的表

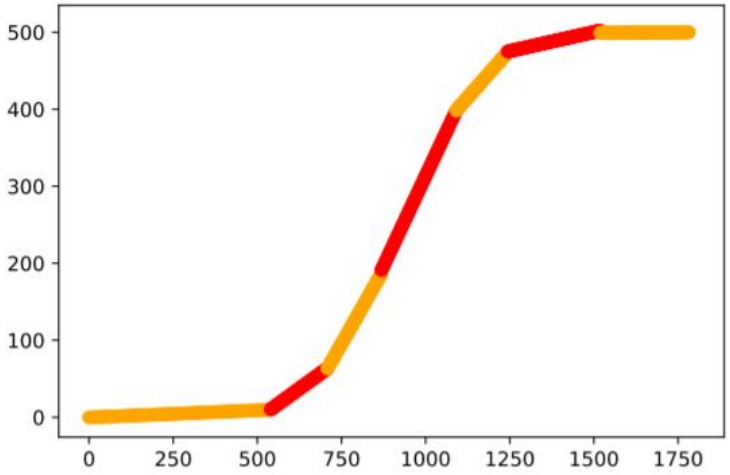
Learning Guidelines

- 在SSTables的查询次数
  - 一个模型的有用频率
- 在SSTable查询的次数由多种因素影响
  - 负载的分布
  - 数据载入的顺序
  - 更高level的文件可能会处理多次内部的查询
- 学习指导3:不要忽略更高level的表
- 学习指导4:对负载和数据敏感

Learning Algorithm:Greedy-PLR

Greedy Piecewise Linear Regression

- From Dataset D
- Multiple linear segments $f(\cdot)$
- $\forall (x, y) \in D, |f(x) - y| < error$
- $error$ is specified beforehand
- In bourbon, we set $error = 8$



Train complexity: O(n)

- Typically ~40ms

Inference complexity: O(log #seg)

- Typically <1μs

Learning Algorithm:Greedy-PLR

- Greedy-PLR一次处理一个数据点
  - 数据点在不违反误差范围小,如果不能加入到当前的line segment
  - 新生成一个line segment,然后将数据点加入到其中

- 模型训练好后,寻找数据很方便
  - 假设包含这个key的line segment找到了,直接可以从这个segment找到key
  - 假设key不在预测的位置,在误差范围决定的区域本地搜索这个key
    - 时间复杂度:O(log-s),s是segment的数量+本地搜索的常数时间复杂度

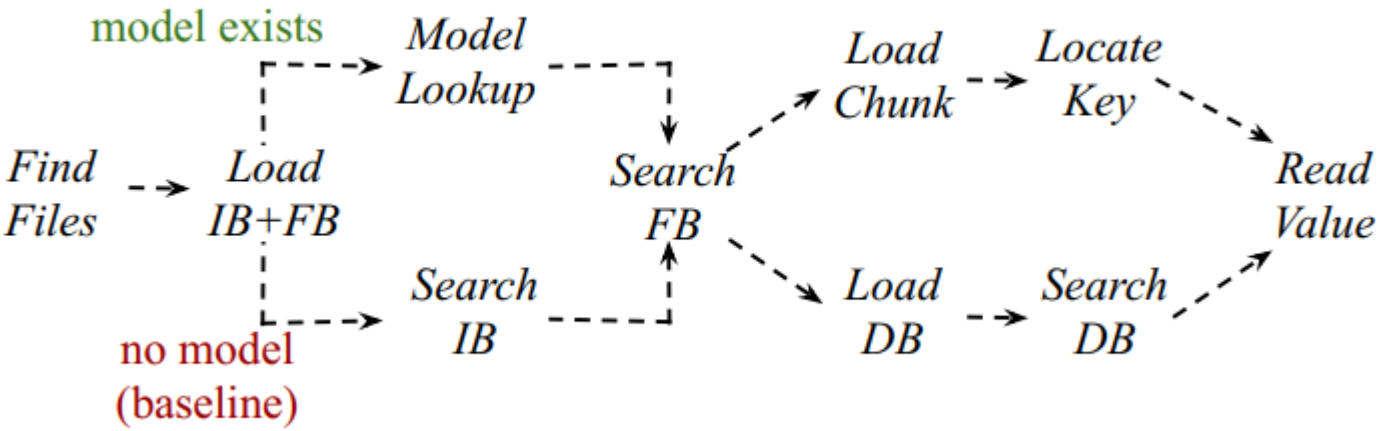
Level vs File Learning

- 一个问题:level learning or file learning?
  - 在write-heavy的负载下,file的模型比level生存更久

Workload	Baseline time (s)	File model		Level model	
		Time(s)	% model	Time(s)	% model
Mixed: Write-heavy	82.6	71.5 (1.16 ×)	74.2	95.1 (0.87 ×)	1.5
Mixed: Read-heavy	89.2	62.05 (1.44 ×)	99.8	74.3 (1.2 ×)	21.4
Read-only	48.4	27.2 (1.78 ×)	100	25.2 (1.92 ×)	100

Table 1: **File vs. Level Learning.** *The table compares the time to perform 10M operations in baseline WiscKey, file-learning, and level-learning. The numbers within the parentheses show the improvements over baseline. The table also shows the percentage of lookups that take the model path; remaining take the original path because the models are not rebuilt yet.*

Bourbon: Putting it All Together



(a) Lookup paths

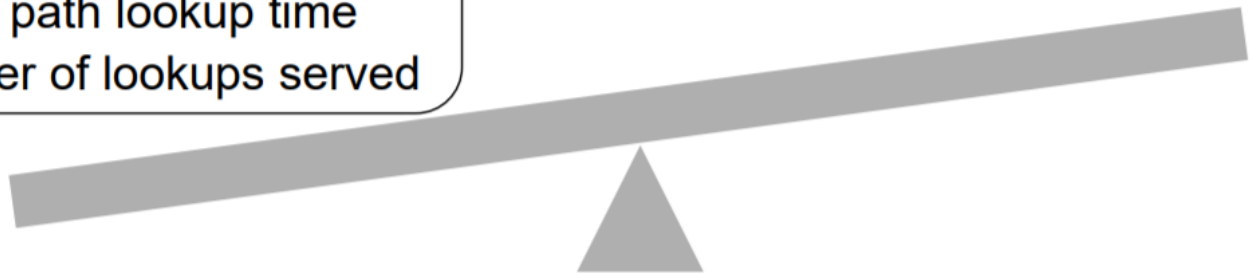
Cost-Benefit Analyzer

- 目标:减少总共的CPU时间
  - trade-off: always learn and no-learn

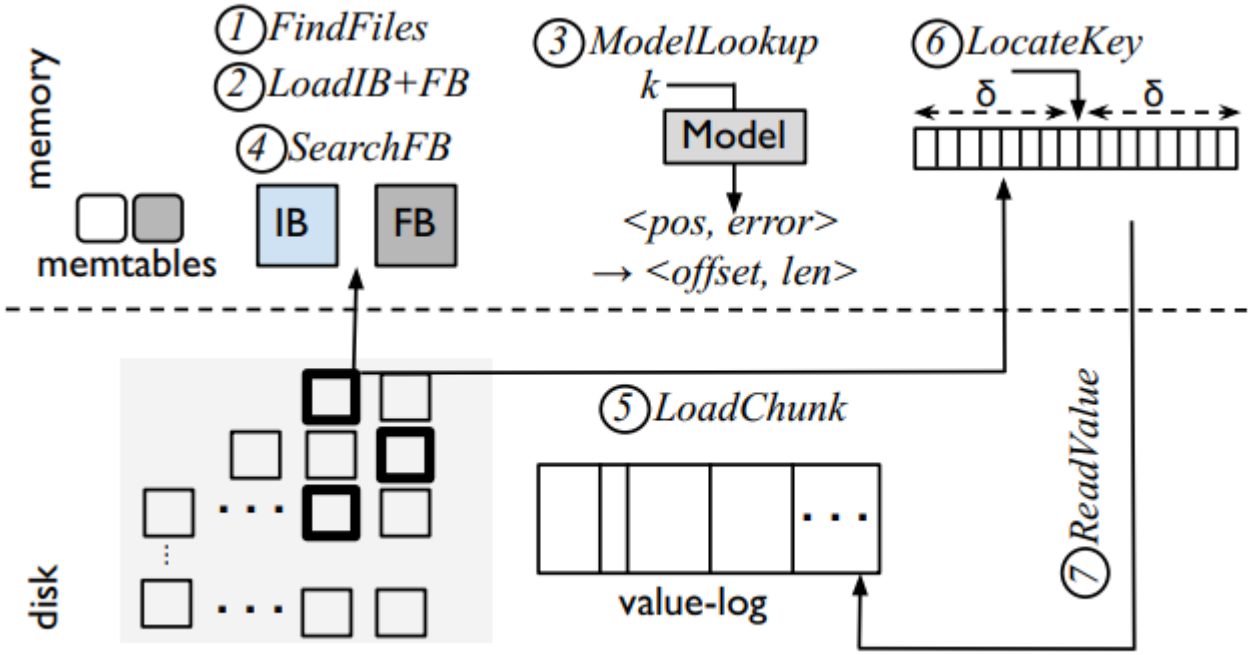
Learn!

Estimated benefit
- Baseline path lookup time
- Model path lookup time
- Number of lookups served

Estimated cost
- Table size



Bourbon: Putting it All Together



(b) Lookup via model - detailed steps