DASYA www.dasya.dk @dasyaITU

DAPHNE https://daphne-eu.github.io/

RAD https://rad.itu.dk

www.itu.dk

micro-architectural analysis of a learned index

Mikkel Møller Andersen & Pınar Tözün

IT University of Copenhagen

pito@itu.dk, www.pinartozun.com, @pinartozun

learned indexes

SIGMOD'18

The Case for Learned Index Structures

Tim Kraska* MIT kraska@mit.edu
Alex Beutel Google, Inc. abeutel@google.com
Ed H. Chi Google, Inc. edchi@google.com
Jeffrey Dean Google, Inc. jeff@google.com
Neoklis Polyzotis Google, Inc. npoly@google.com

SIGMOD'19

FITing-Tree: A Data-aware Index Structure

Alex Galakatos[1*] Michael Markovitch[1*] Carsten Binnig[2]
Rodrigo Fonseca[1] Tim Kraska[3]
[1]Brown University {first_last}@brown.edu [2]TU Darmstadt {first.last@cs.tu-darmstadt.de} [3]MIT CSAIL {last@mit.edu}

PVLDB'20

The PGM-index: a fully-dynamic compressed learned index with provable worst-case bounds

Paolo Ferragina University of Pisa, Italy paolo.ferragina@unipi.it
Giorgio Vinciguerra University of Pisa, Italy giorgio.vinciguerra@phd.unipi.it

EDBT'21

Shift-Table: A Low-latency Learned Index for Range Queries using Model Correction

Ali Hadian Imperial College London
Thomas Heinis Imperial College London

aiDM'20

RadixSpline: A Single-Pass Learned Index

Andreas Kipf★ Ryan Marcus★† Alexander van Renen Mihail Stoian
Alfons Kemper Tim Kraska★ Thomas Neumann
TUM MIT CSAIL★ Intel Labs†
{renen, stoian, kemper, neumann}@in.tum.de {kipf, ryanmarcus, kraska}@mit.edu

SIGMOD'20

ALEX: An Updatable Adaptive Learned Index

Jialin Ding† Umar Farooq Minhas‡ Jia Yu§*
Chi Wang‡ Jaeyoung Do‡ Yinan Li‡ Hantian Zhang∓* Badrish Chandramouli‡
Johannes Gehrke¶ Donald Kossmann‡ David Lomet‡ Tim Kraska†
†Massachusetts Institute of Technology ‡Microsoft Research §Arizona State University
∓Georgia Institute of Technology ¶Microsoft

learned indexes

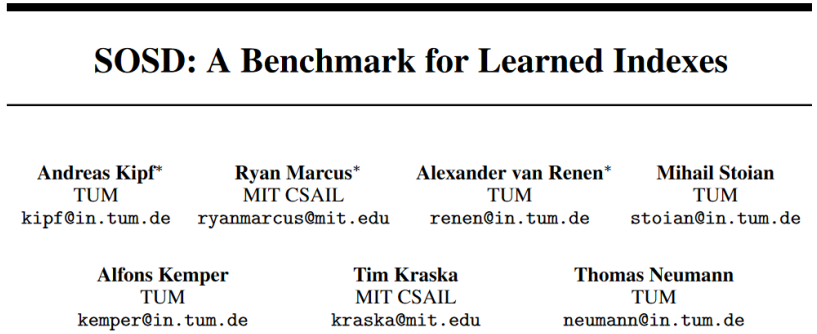
use a model that predicts the position of the tuple given a key

- ✓ trading off increased computation to reduced memory accesses
- ✓ reduced index size

benefits shown by prior work using high level metrics:

- *throughput, latency, index size …*

+ a standardized benchmarking effort



SOSD: A Benchmark for Learned Indexes

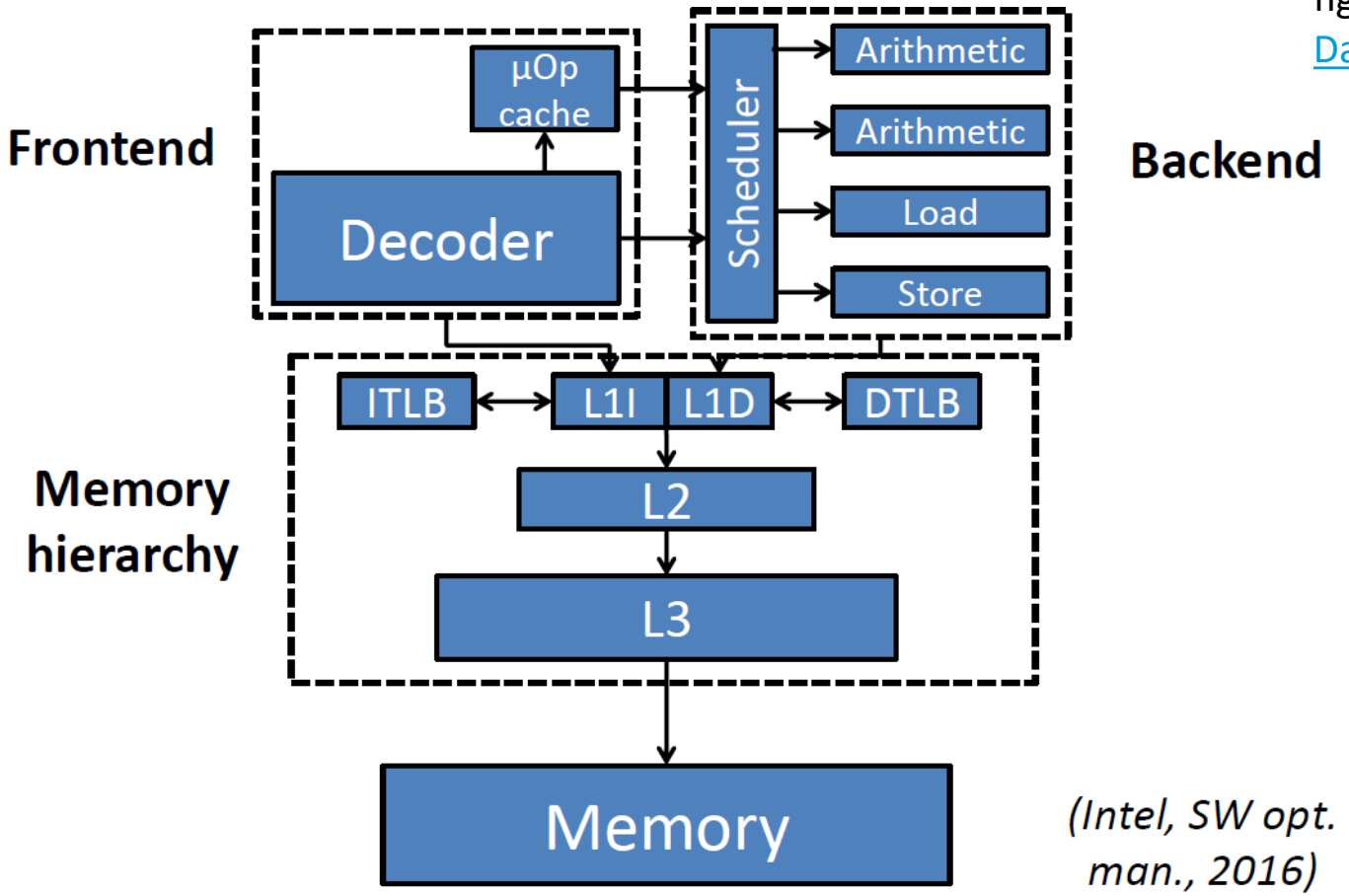
Andreas Kipf* TUM kipf@in.tum.de
Ryan Marcus* MIT CSAIL ryanmarcus@mit.edu
Alexander van Renen* TUM renen@in.tum.de
Mihail Stoian TUM stoian@in.tum.de
Alfons Kemper TUM kemper@in.tum.de
Tim Kraska MIT CSAIL kraska@mit.edu
Thomas Neumann TUM neumann@in.tum.de

how about lower-level metrics?

where does time go at a micro-architectural level?

micro-architecture of an OoO processor core

figure from Utku Şirin's DaMoN 2017 presentation



(Intel, SW opt. man., 2016)

delays in fetching, decoding, etc. an instruction cause *frontend stalls*, rest cause *backend stalls*

micro-architectural analysis

what is the time-breakdown across micro-architectural components?

- helps us understand how a processor's resources are utilized
- hints what can be improved both from the hardware or software side

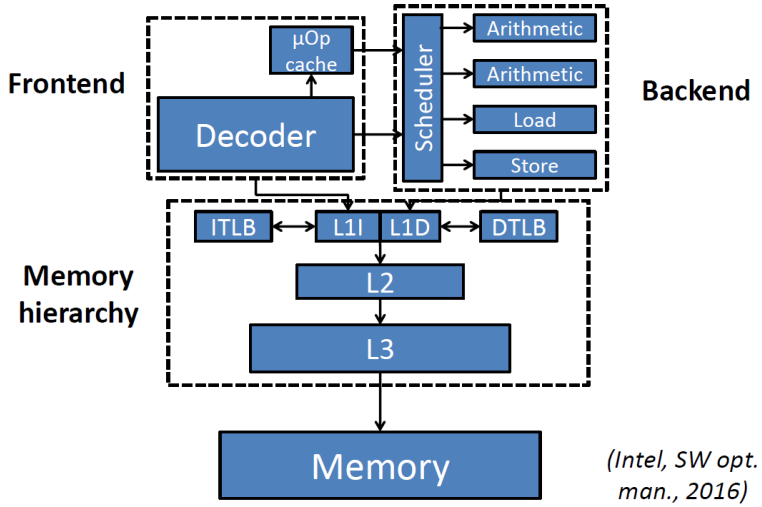
DBMSs On A Modern Processor: Where Does Time Go?

VLDB'99

Anastassia Ailamaki David J. DeWitt Mark D. Hill David A. Wood
University of Wisconsin-Madison

measured using hardware counters

- popular tools:
  perf (generic linux),
  VTune (for Intel hardware)



(Intel, SW opt. man., 2016)

figure courtesy of Utku Şirin

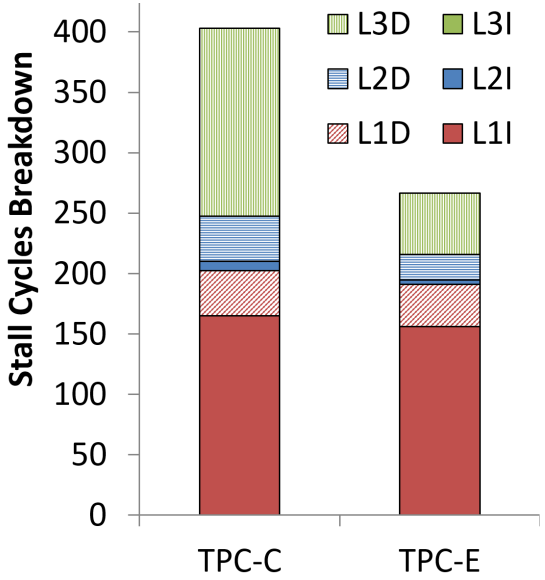
micro-architectural analysis

back in the old days

- *# cache misses from Ln X*
  expected latency for cache misses from Ln

➔ gives a good estimate

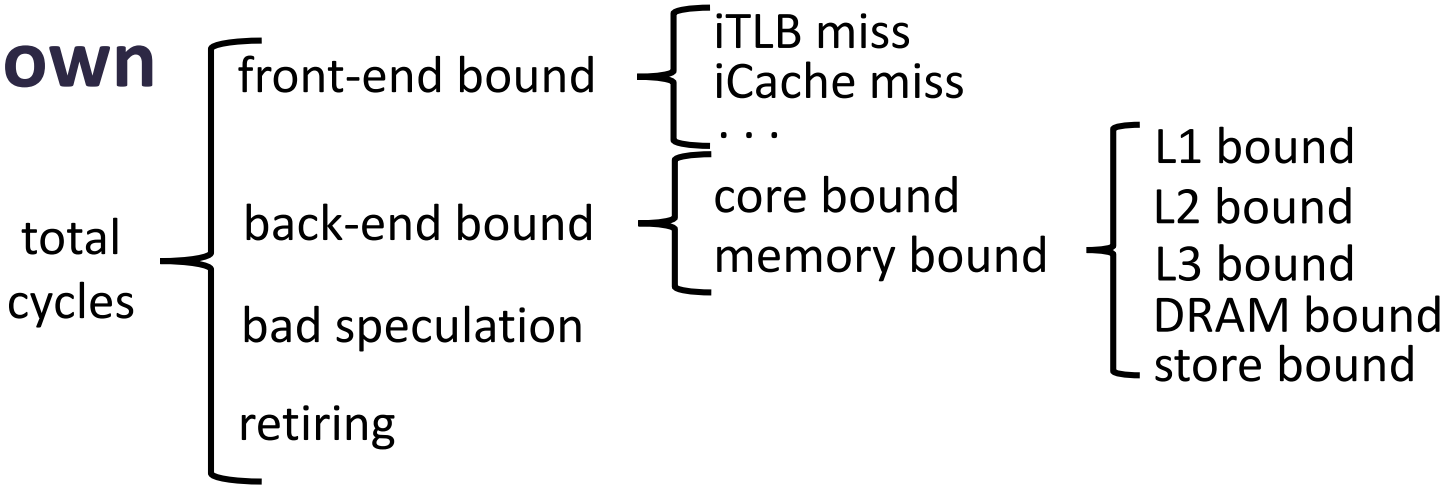
➔ doesn't account for overlaps in cycles



today

- Intel's TMAM (Top-Down Micro-architecture Analysis Method)

➔a more precise breakdown



experimental setup

- software: ALEX, ART, B+tree
- hardware: Intel Xeon Platinum 8256

Intel DevCloud

- workload:
  - point queries
    - read-only = 100% reads
    - read-heavy = 80% reads – 20% updates/inserts
    - write-heavy = 40% reads – 60% updates/inserts/deletes
    - insert-only = 100% inserts
  - dataset: uniform random 8byte key-value pairs
  - insert patterns: random & consecutive
  - #keys at start: 1.6 million, 1.6 billion
- runs: (1) data population, (2) warm-up, (3) actual run
- TMAM with VTune

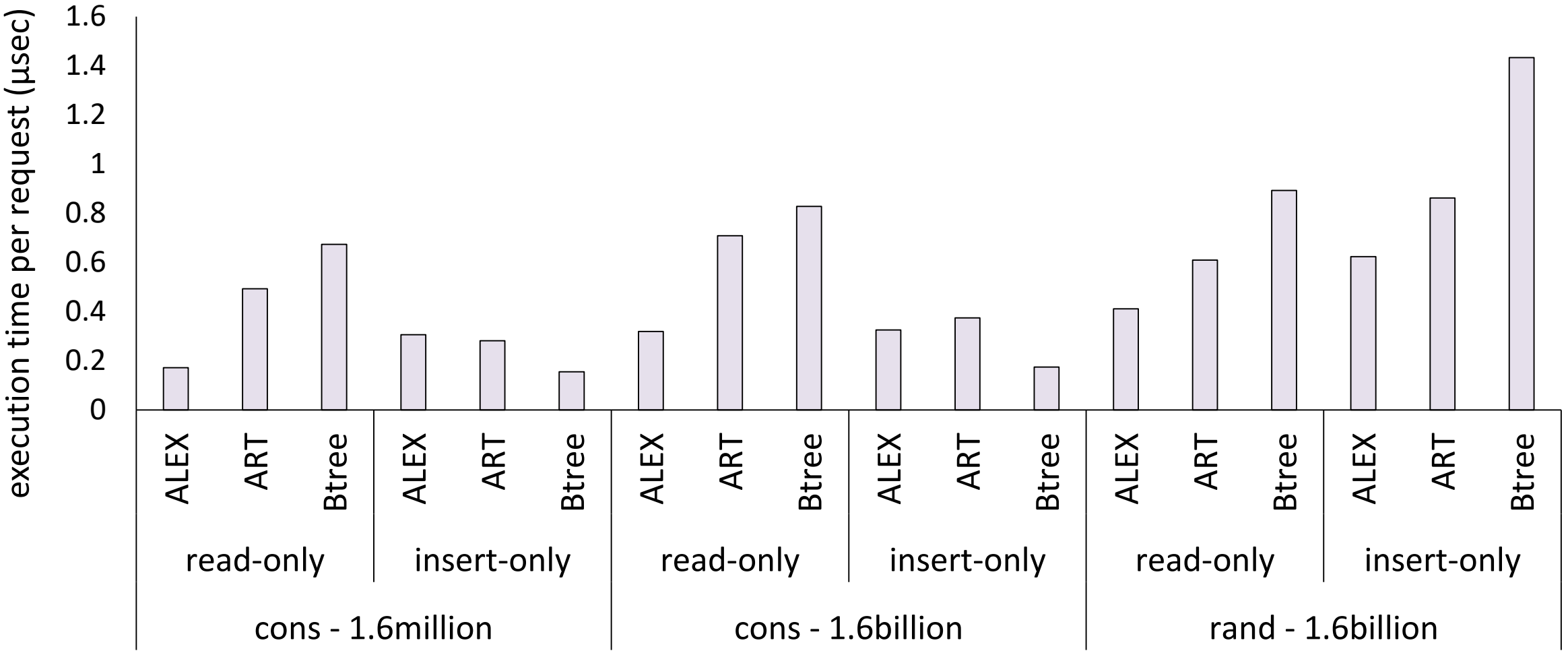
SIGMOD'20

ALEX: An Updatable Adaptive Learned Index

Jialin Ding[†] Umar Farooq Minhas[‡] Jia Yu[§*]
Chi Wang[‡] Jaeyoung Do[‡] Yinan Li[‡] Hantian Zhang[∓*] Badrish Chandramouli[‡]
Johannes Gehrke[¶] Donald Kossmann[‡] David Lomet[‡] Tim Kraska[†]
[†]Massachusetts Institute of Technology [‡]Microsoft Research [§]Arizona State University
[∓]Georgia Institute of Technology [¶]Microsoft

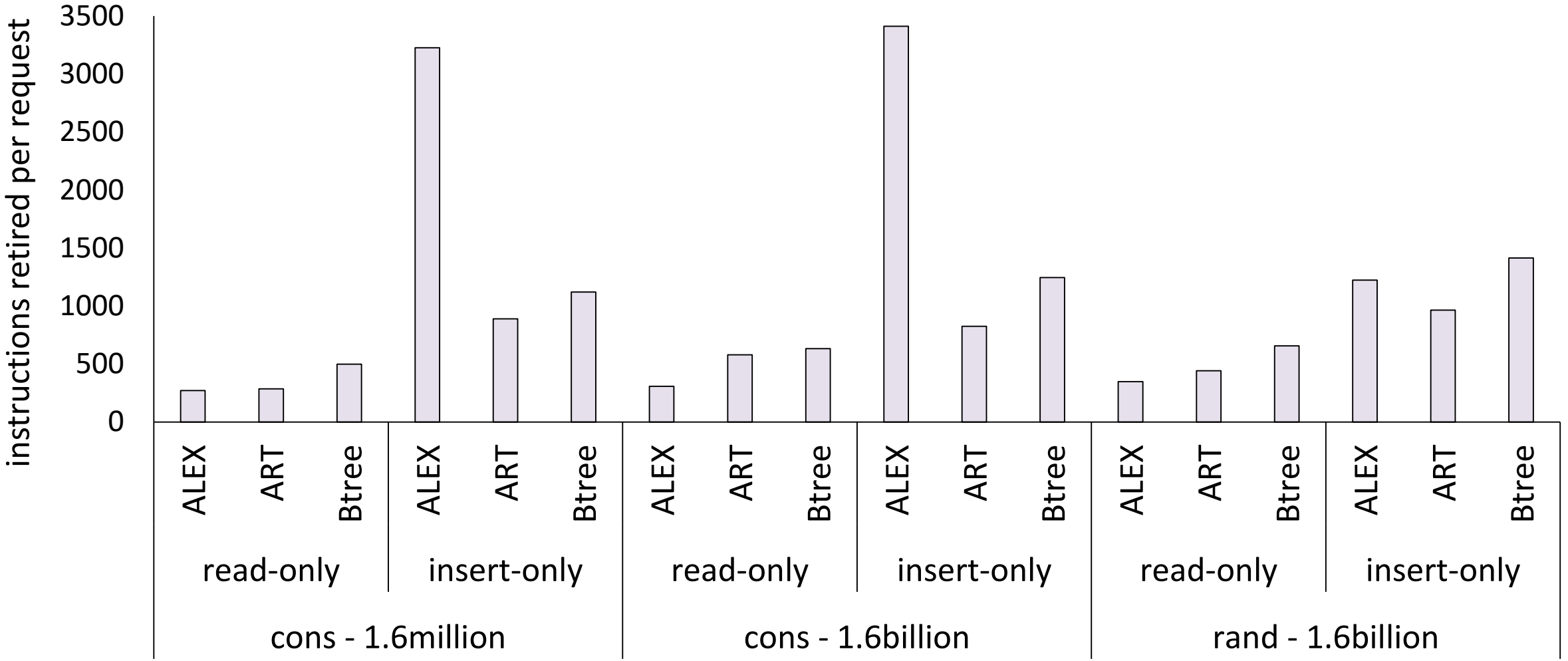
does micro-architectural behavior of learned indexes differ from more traditional ones?

execution time



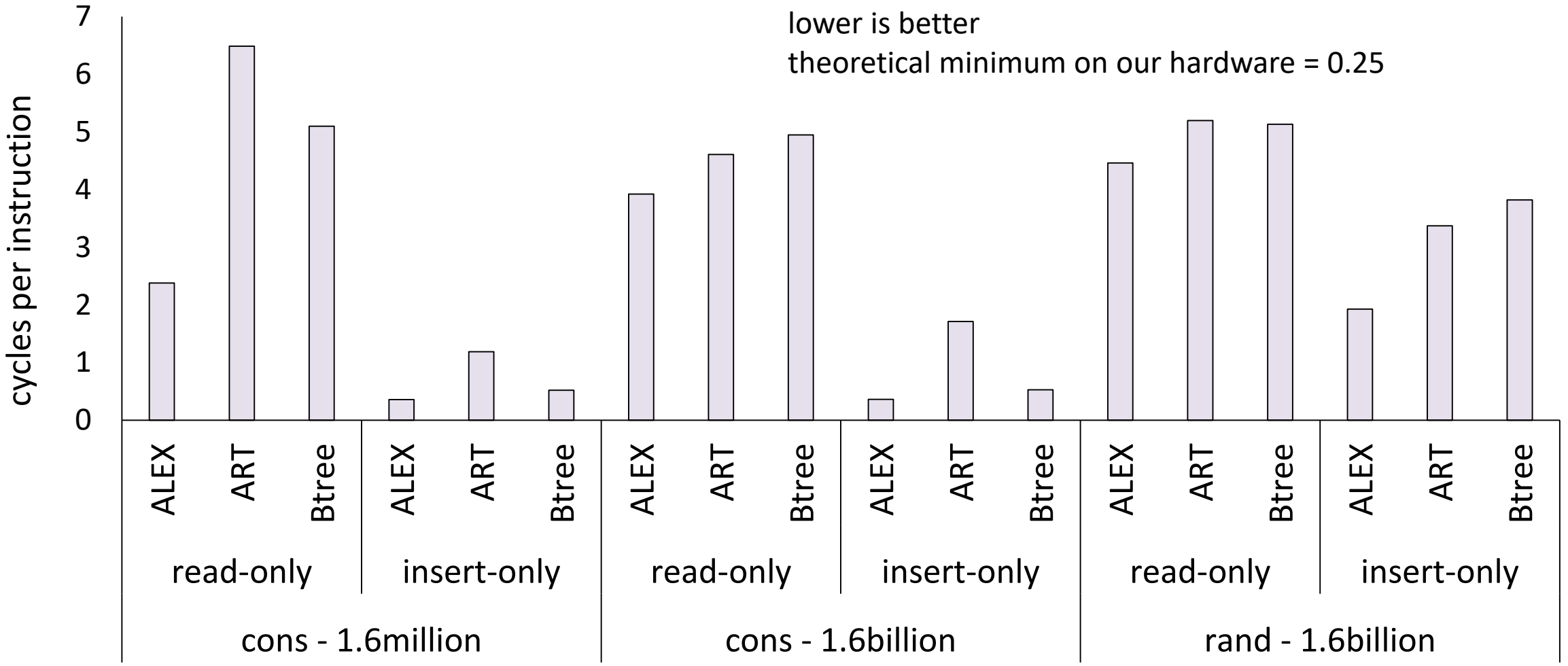
ALEX outperforms others in general except for heavy consecutive inserts

instruction footprint



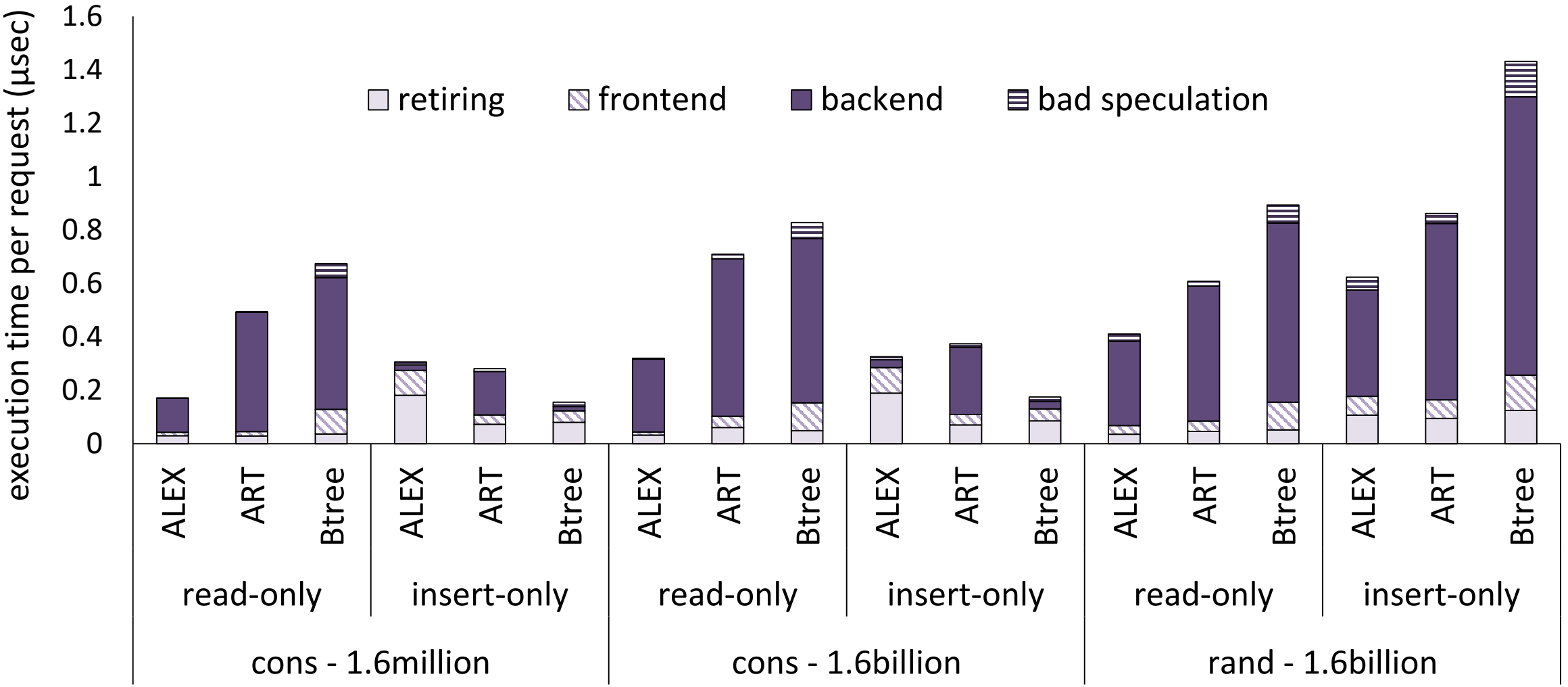
consecutive inserts drastically increase the instruction footprint per request for ALEX

instruction-level parallelism (ILP)



consecutive inserts increase ILP, especially for ALEX

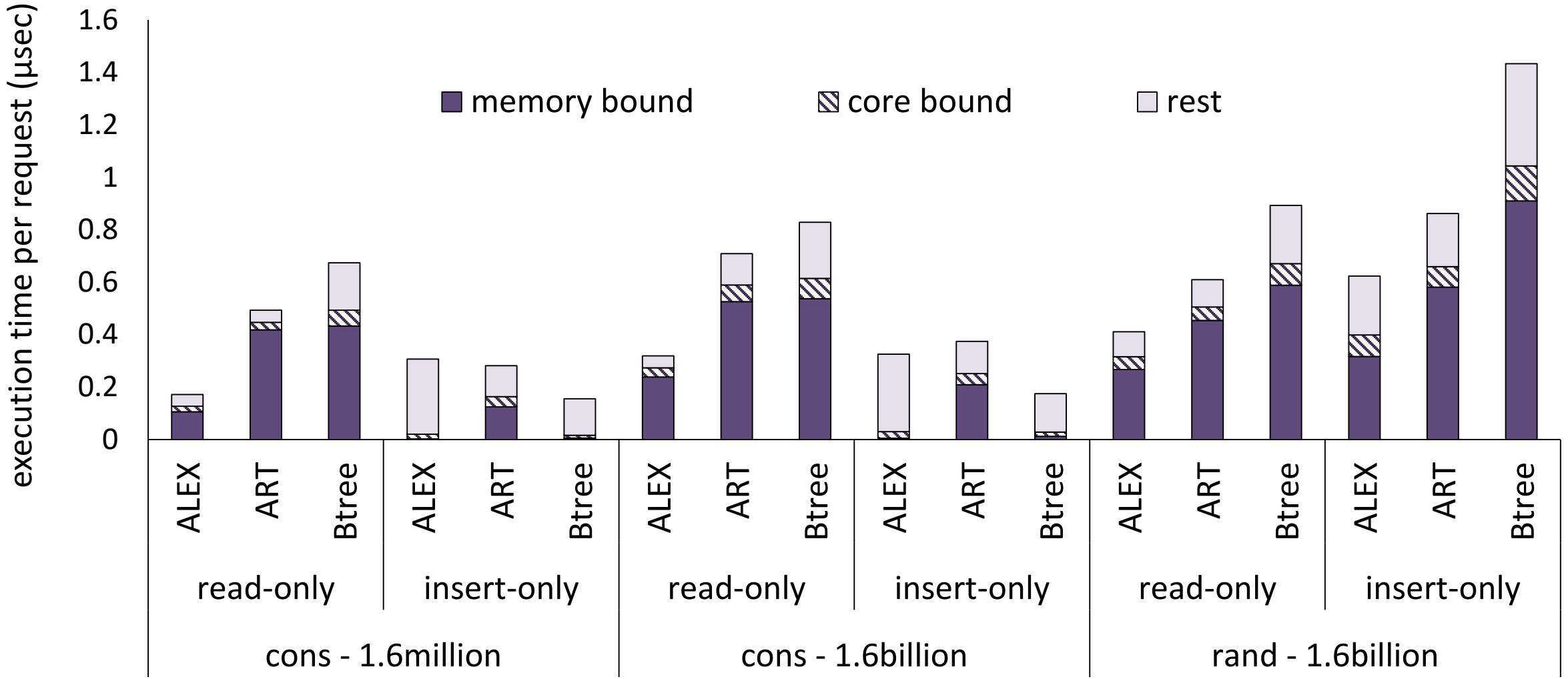
breakdown of execution time



backend stalls dominate the execution cycles

frontend isn't bottlenecked

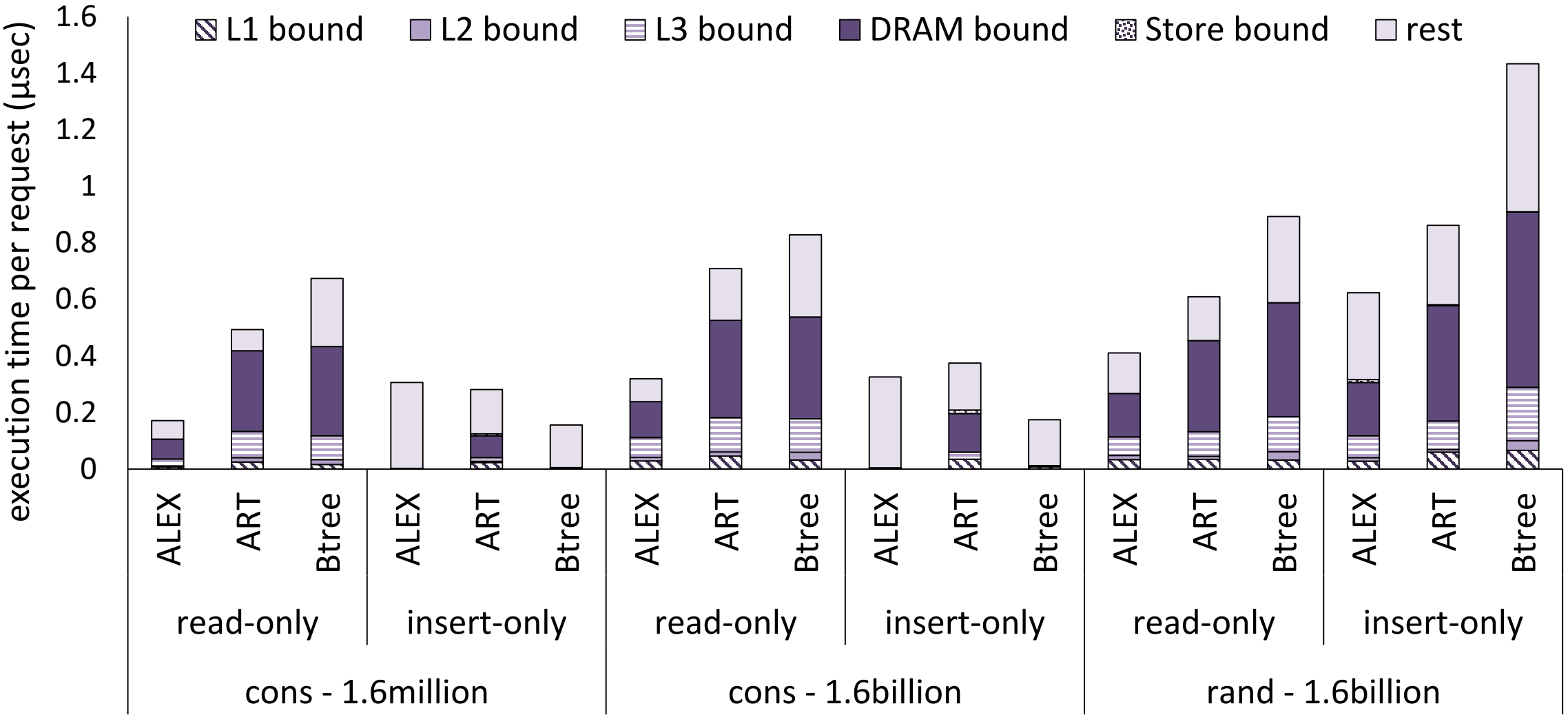
breakdown of backend stalls



memory stalls dominate the backend cycles

expected behavior for most data-intensive applications

breakdown of memory stalls



long-latency data misses dominate the memory stalls
expected behavior for most data-intensive applications

conclusion

tools for micro-architectural analysis have evolved tremendously

➔ **for work that deals with data structures,**
consider adding results with lower-level metrics

highlights of results:

➔ **all indexes are bound by long-latency cache misses**

➔ **learned indexes may exhibit high instruction footprint (e.g. insert-heavy)**
but also high instruction-level parallelism

future directions:

- analysis with SOSD benchmark (non-uniform cases)
- impact of multi-threading
- per request analysis
- analysis within a database / data management system

thank you!