Interleaving with Coroutines

A Practical Approach to Avoid Memory Stalls

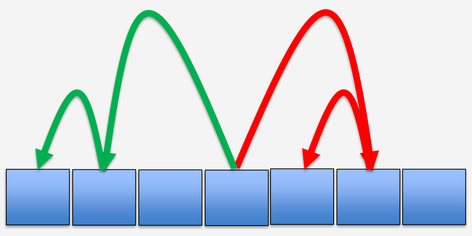
Georgios Psaropoulos*†, Thomas Legler†, Norman May†, Anastasia Ailamaki*

*EPFL, †SAP SE

1. Index joins in database systems

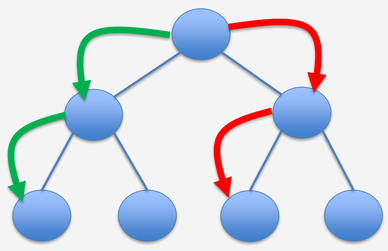
- Databases use index structures for random data access:

Sorted arrays with binary search



➔Indirect memory access

Tree structures



➔Pointer chasing

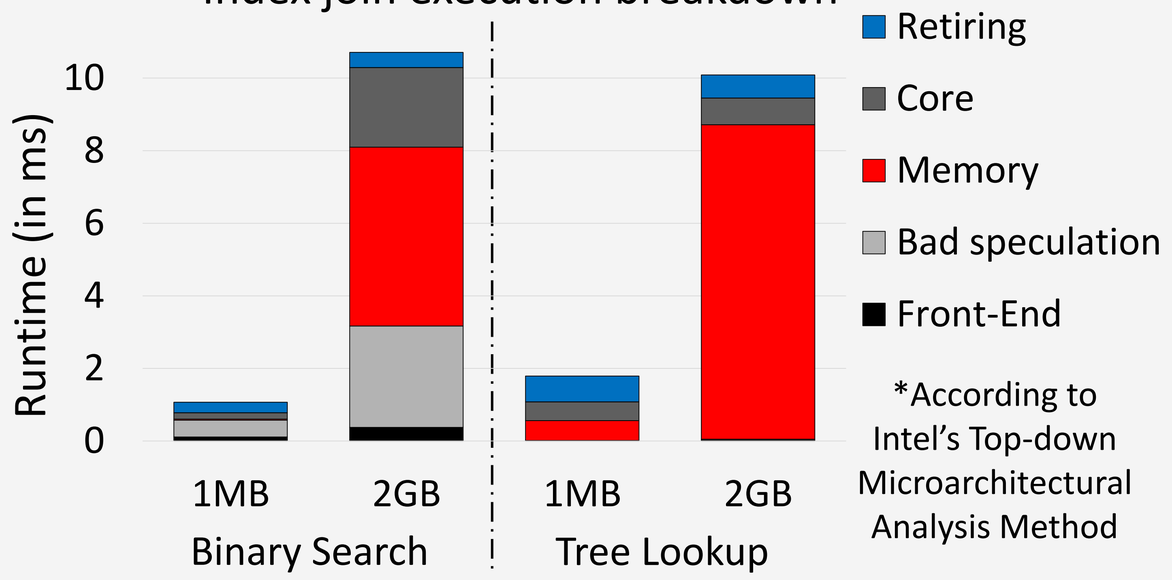
- Index lookups → irregular memory access patterns
- Index join: a sequence of independent index lookups

Irregular memory accesses + independent lookups

2. Irregular memory access → wasted cycles

- Index lookups are sensitive to index size
- …regardless of the index structure

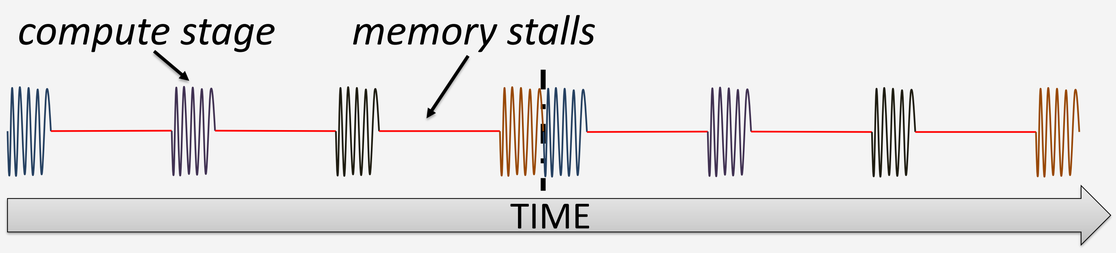
Index join execution breakdown*



Memory stalls: up to 85% of total cycles

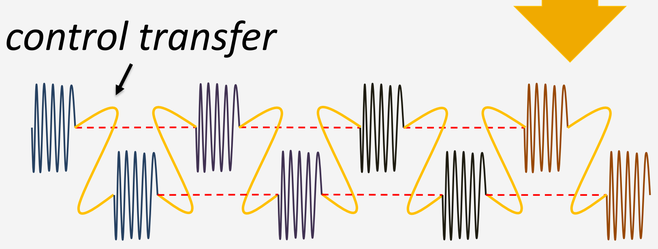
3. Interleaved execution of lookups

Consecutive lookups with 3 cache misses each



Interleaved Execution

Overlap memory access with independent instructions from other lookups



Transfer execution control upon cache miss

Execute independent instructions instead of stalling

4. Interleaved execution with coroutines

Coroutines: functions that suspend and resume their execution

- Binary search as a coroutine:

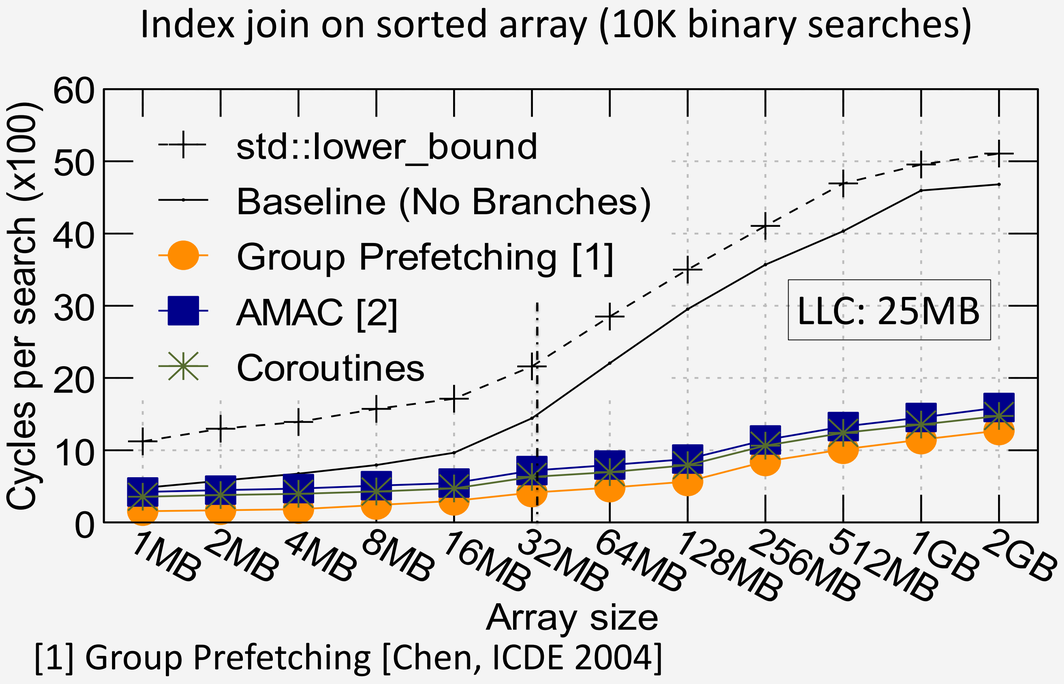
```
   coroutine lookup(array, value):
1.   size = array.size(), low = 0
2.   while size >= 2 do
3.     half = size / 2
4.     probe = low + half
5.     size = half
6.     prefetch(&array[probe])
7.     suspend()
8.     v = array[probe]
9.     low = v < value ? probe : low
10.  return low
```

original cache miss

Minimal and non-intrusive code changes

5. Interleaved vs non-interleaved execution

Index join on sorted array (10K binary searches)



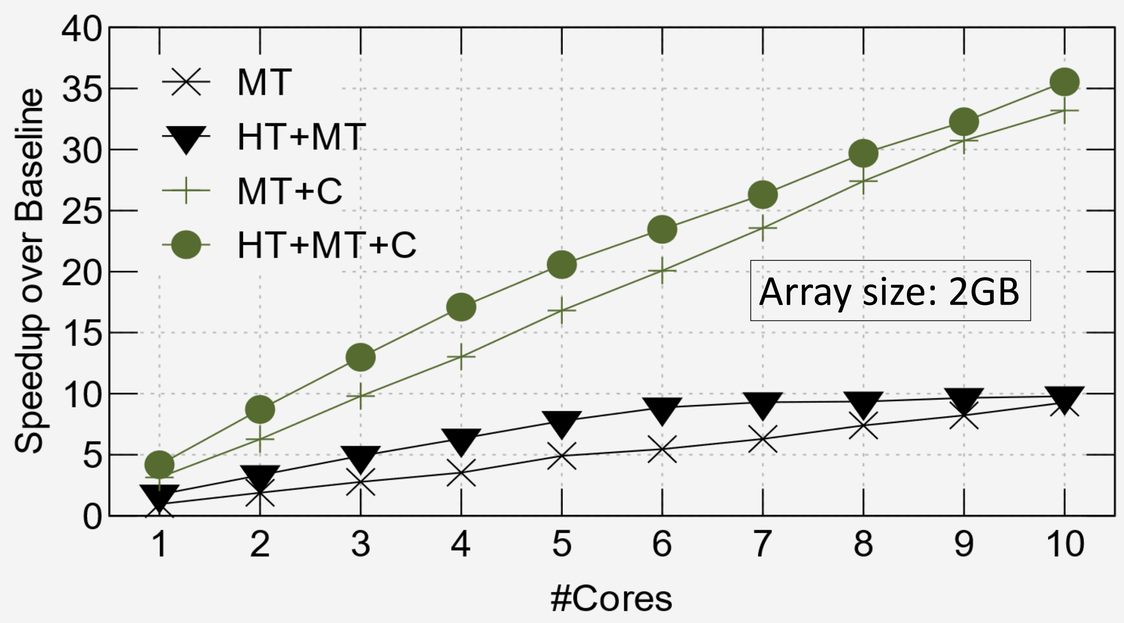
[1] Group Prefetching [Chen, ICDE 2004]

[2] Asynchronous Memory Access Chaining [Kocberber, PVLDB 9(4)]

Interleaved execution → runtime oblivious to array size

6. Multithreaded interleaved execution

Multithreaded (MT) index join on sorted array (10K binary searches)
+ Hyperthreading (HT) + Interleaved Execution (C)



Interleaved execution scales better

Parts of this work have been published at PVLDB 11(2) under the title
"Interleaving with Coroutines: A Practical Approach for Robust Index Joins"