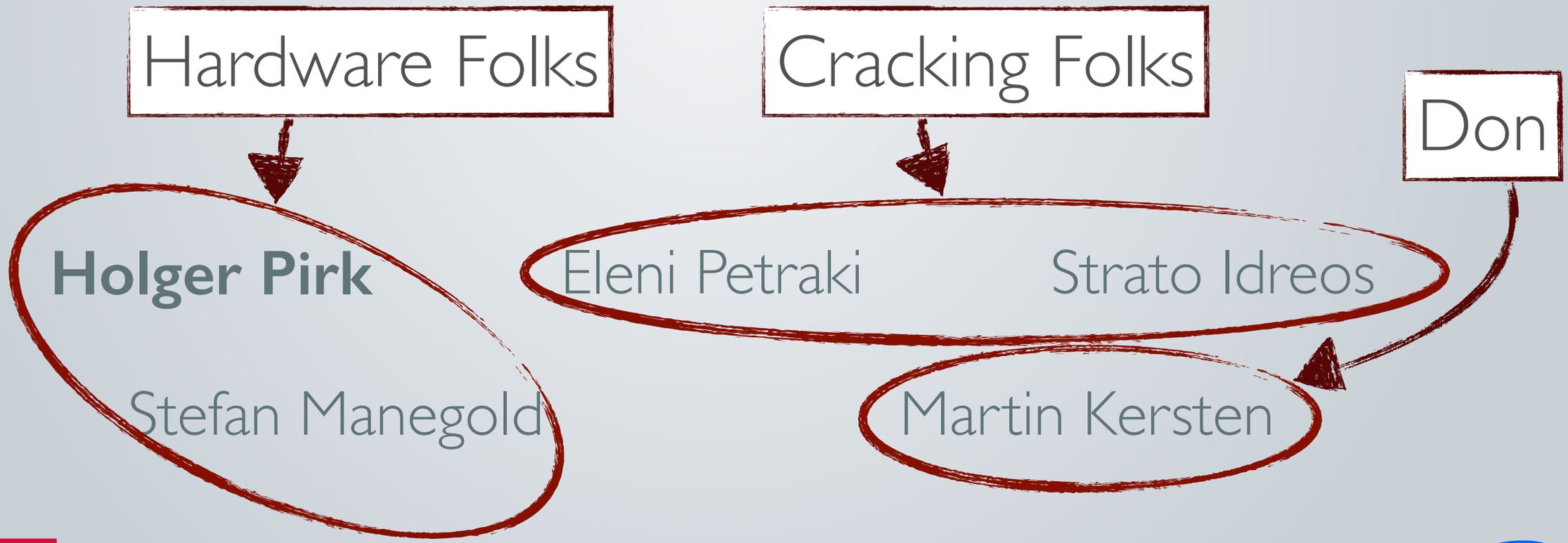
DATABASE CRACKING:

Fancy Scan, not Poor Man's Sort!



CWI

monetdb

EVALUATING RANGE PREDICATES

CWI

monetdb

COMPLEXITY ON PAPER

- Scanning: *O(n)*
- Sorting: *O(n×log(n))*
- Cracking: *O(n)*
  - Essentially a single Quicksort-Step

CWI

monetdb

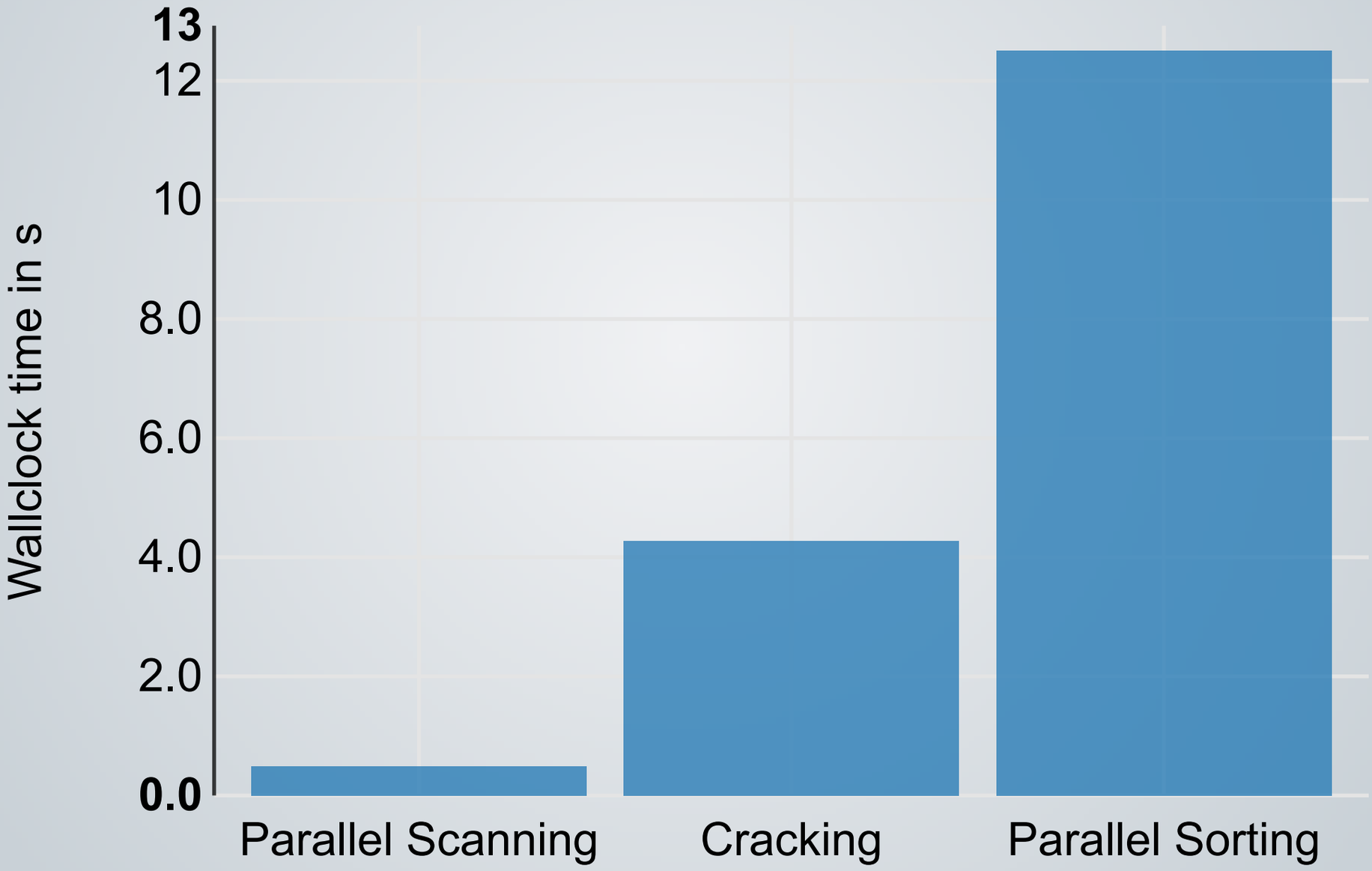
COSTS IN REALITY

- Implement microbenchmarks
- 1 Billion uniform random integer values
- Pivot in the middle of the range
- Workstation machine (16 GB RAM, 4 Sandy Bridge Cores)

CWI

monetdb

COSTS IN REALITY



CWI

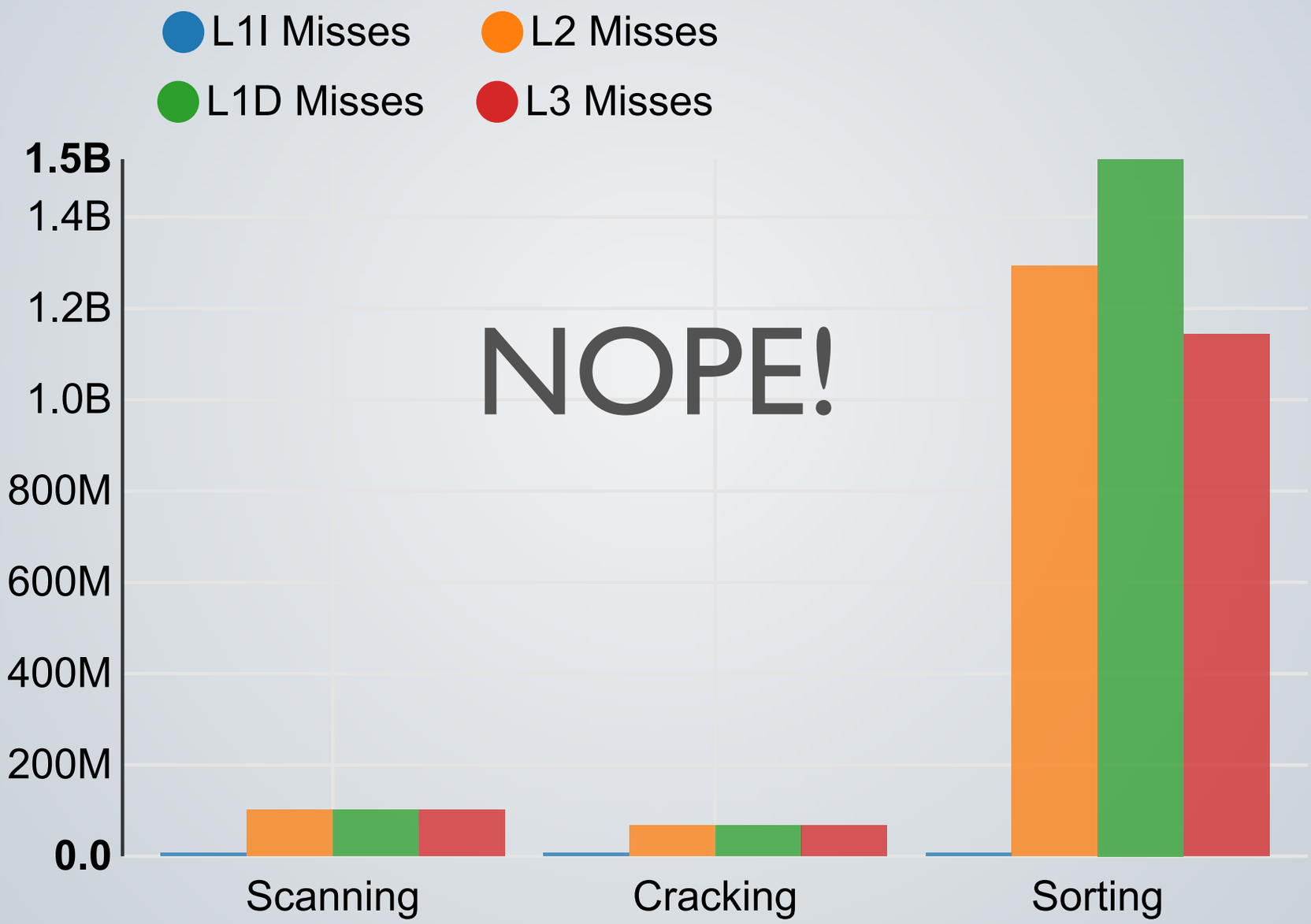
monetdb

SO: WHAT'S GOING ON?

CWI

monetdb

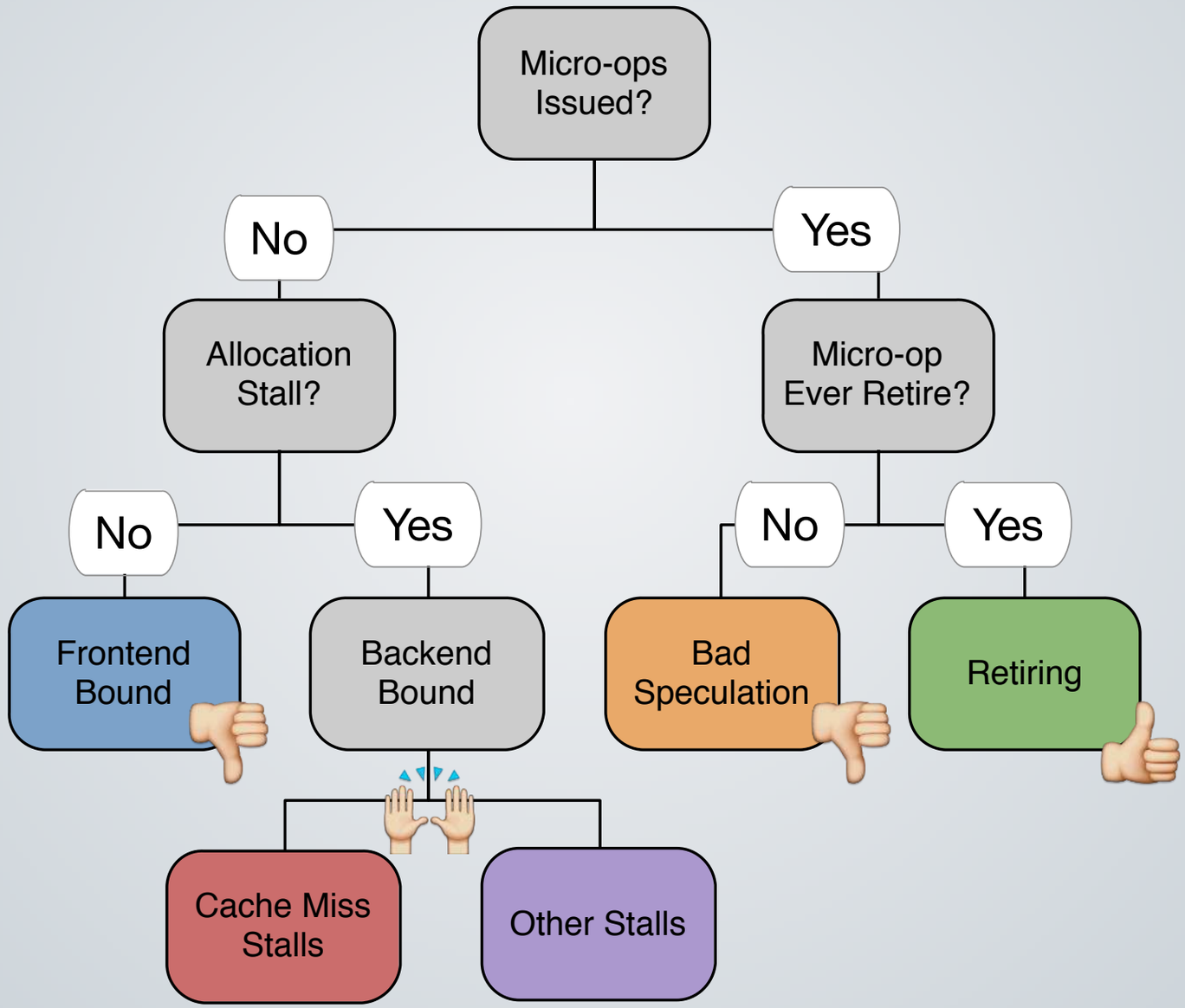
CACHE MISSES?



CWI

monetdb

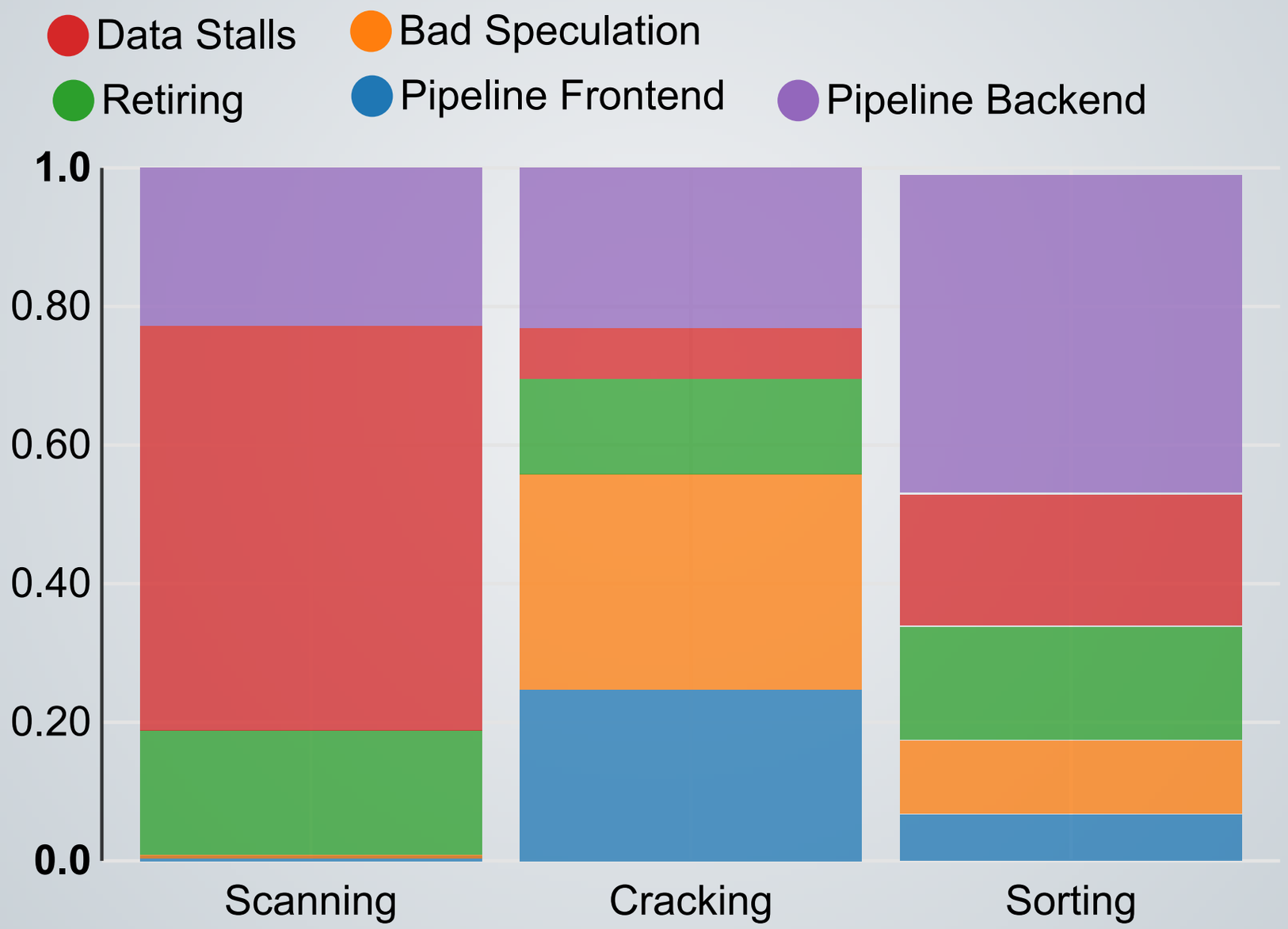
CPU COSTS



CWI

monetdb

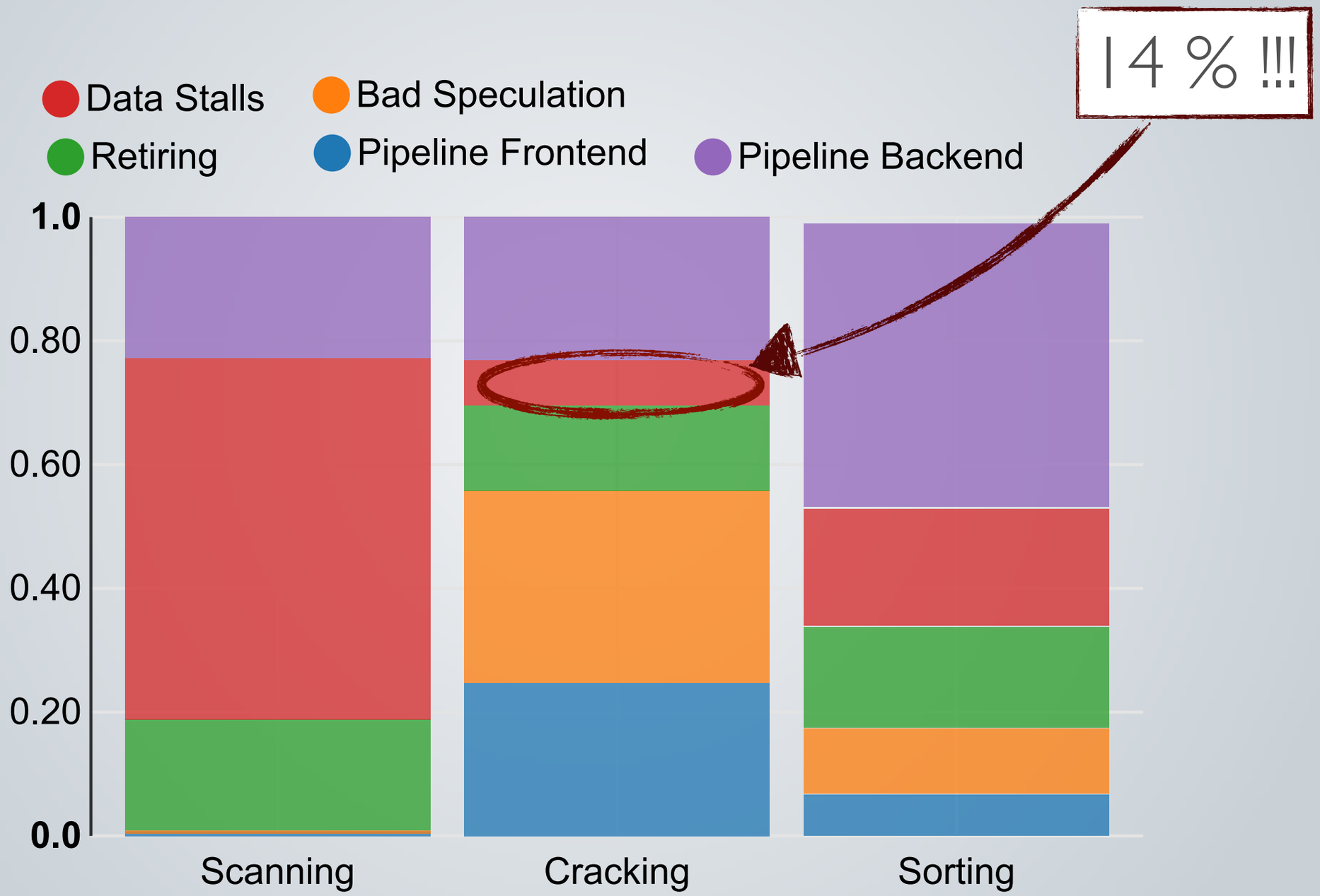
CPU COSTS



CWI

monetdb

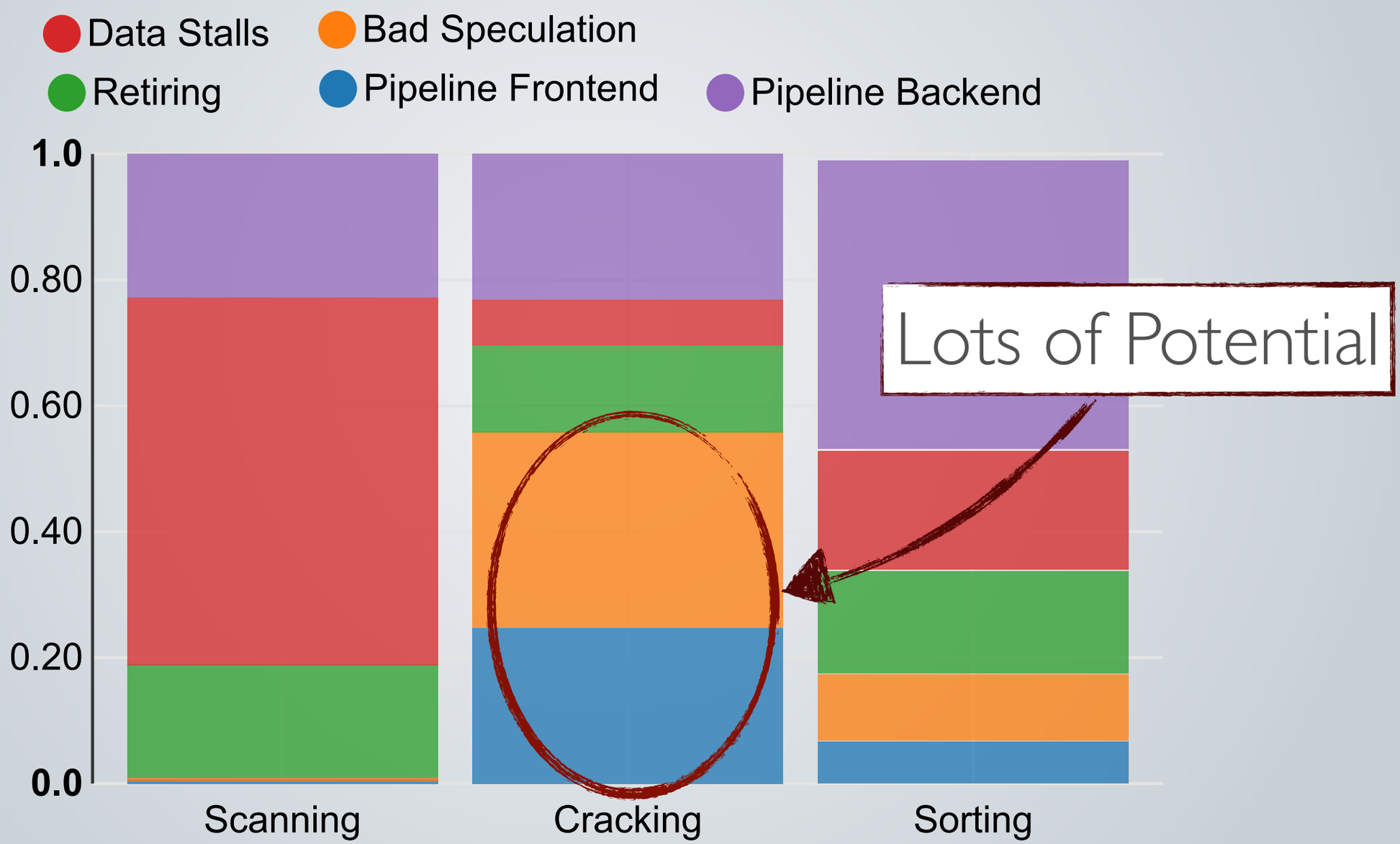
CPU COSTS



CWI

monetdb

CPU COSTS



CWI

monetdb

WHAT CAN WE DO ABOUT IT?

CWI

monetdb

INCREASING CPU EFFICIENCY

CWI

monetdb

PREDICATION

```
for(i=0; i<size; i++)
  if(input[i] < pivot){
    output[outl] = input[i];
    outl++
  }
```

```
for(i=0; i<size; i++)
  {
    output[outl] = input[i];
    outl += (input[i] < pivot);
  }
```

PREDICATION

- Turns control dependencies into data dependencies
- Eliminates Branch Mispredictions
- Causes unconditional (potentially unnecessary) I/O
  - (limited to caches)
- Works only for out-of-place algorithms

PREDICATED CRACKING

CWI

monetdb

PREDICATED CRACKING

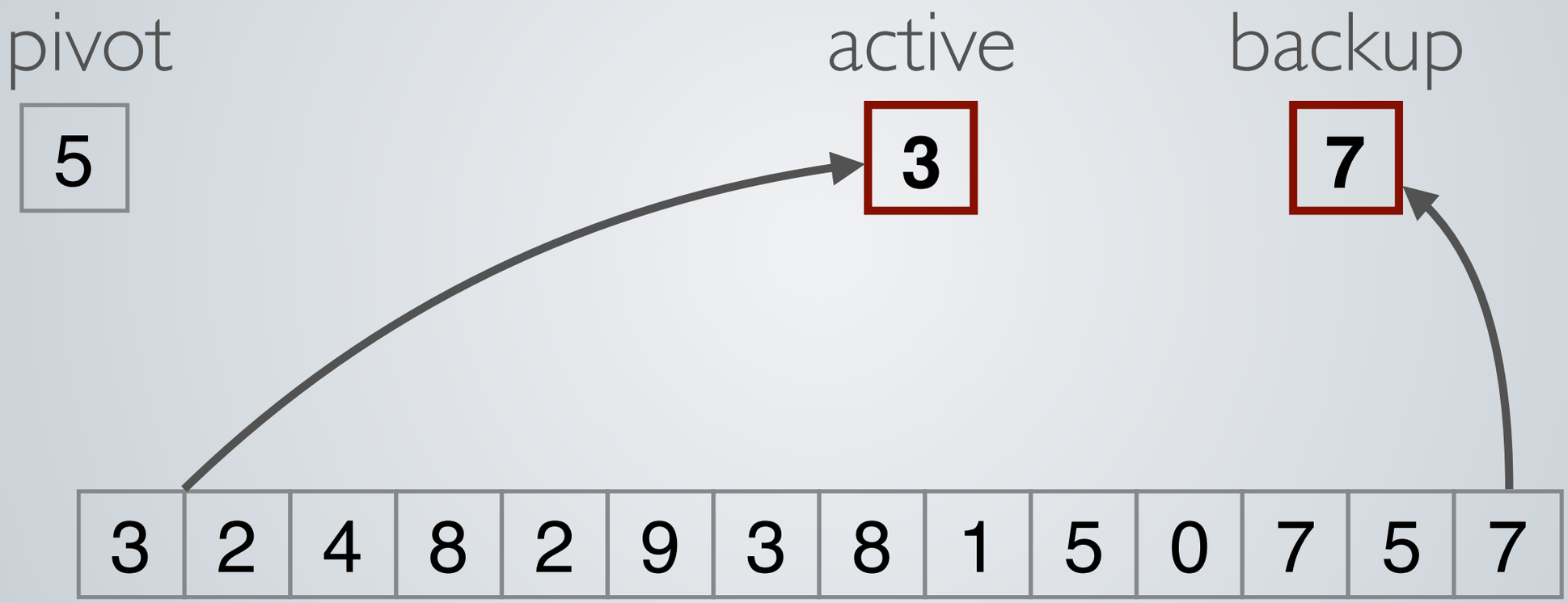
pivot	active	backup
5		

7	2	4	8	2	9	3	8	1	5	0	7	5	3

CWI

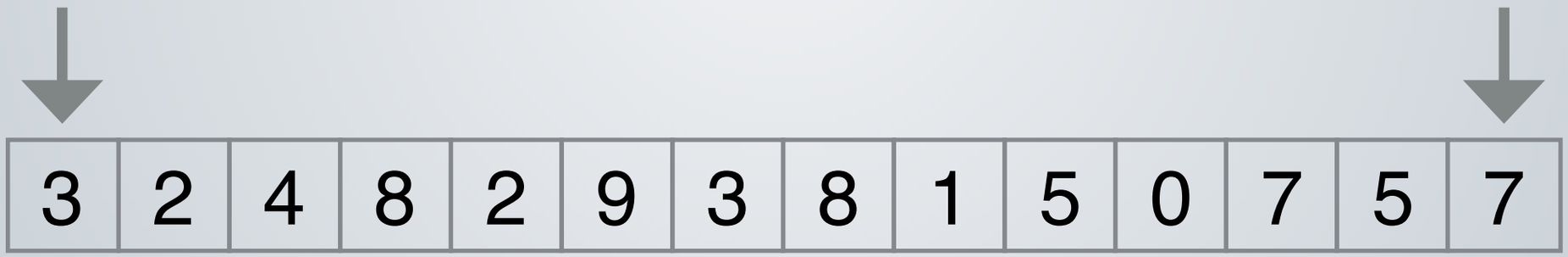
monetdb

PREDICATED CRACKING



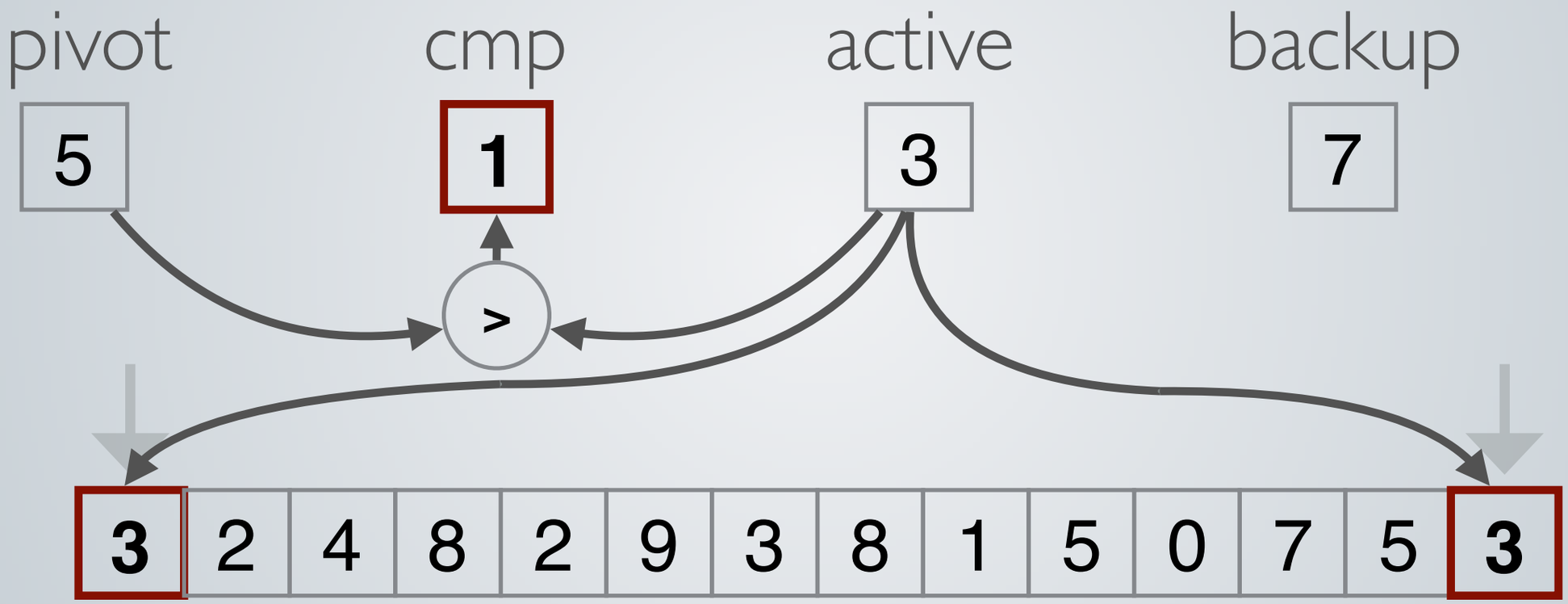
PREDICATED CRACKING

pivot	cmp	active	backup
5	?	3	7



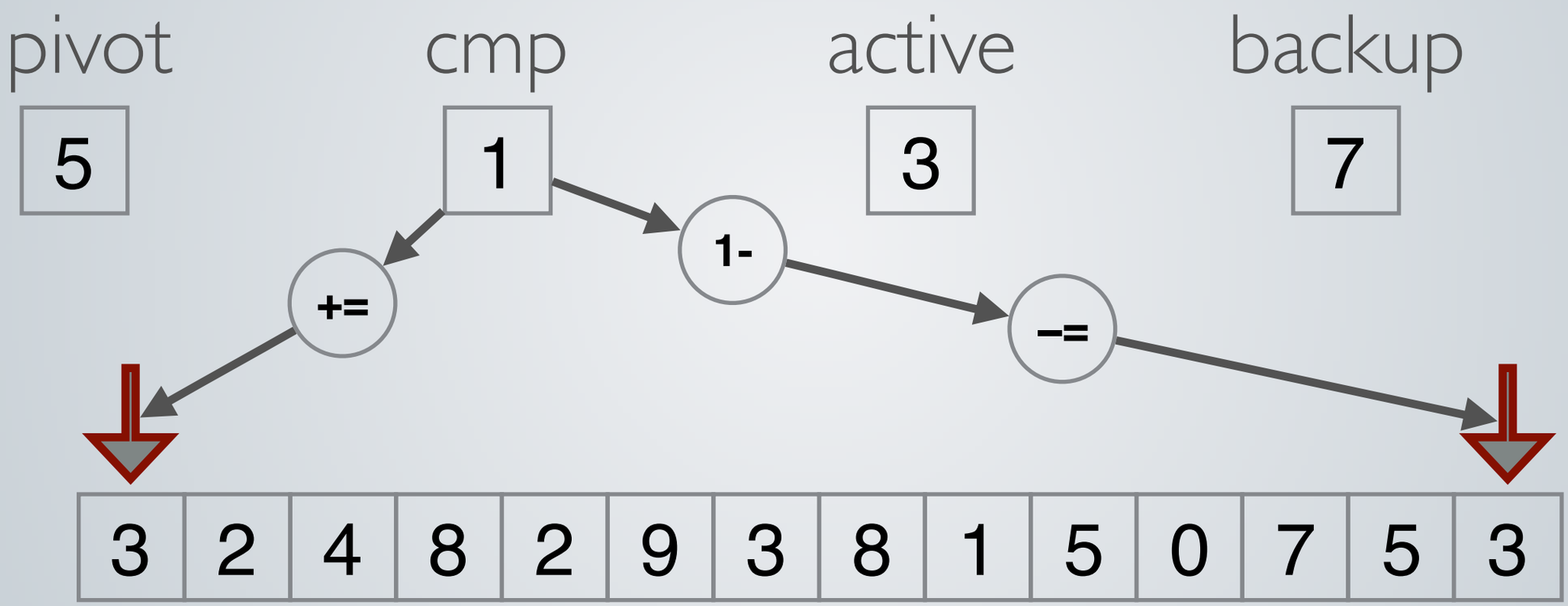
State Before Iteration

PREDICATED CRACKING



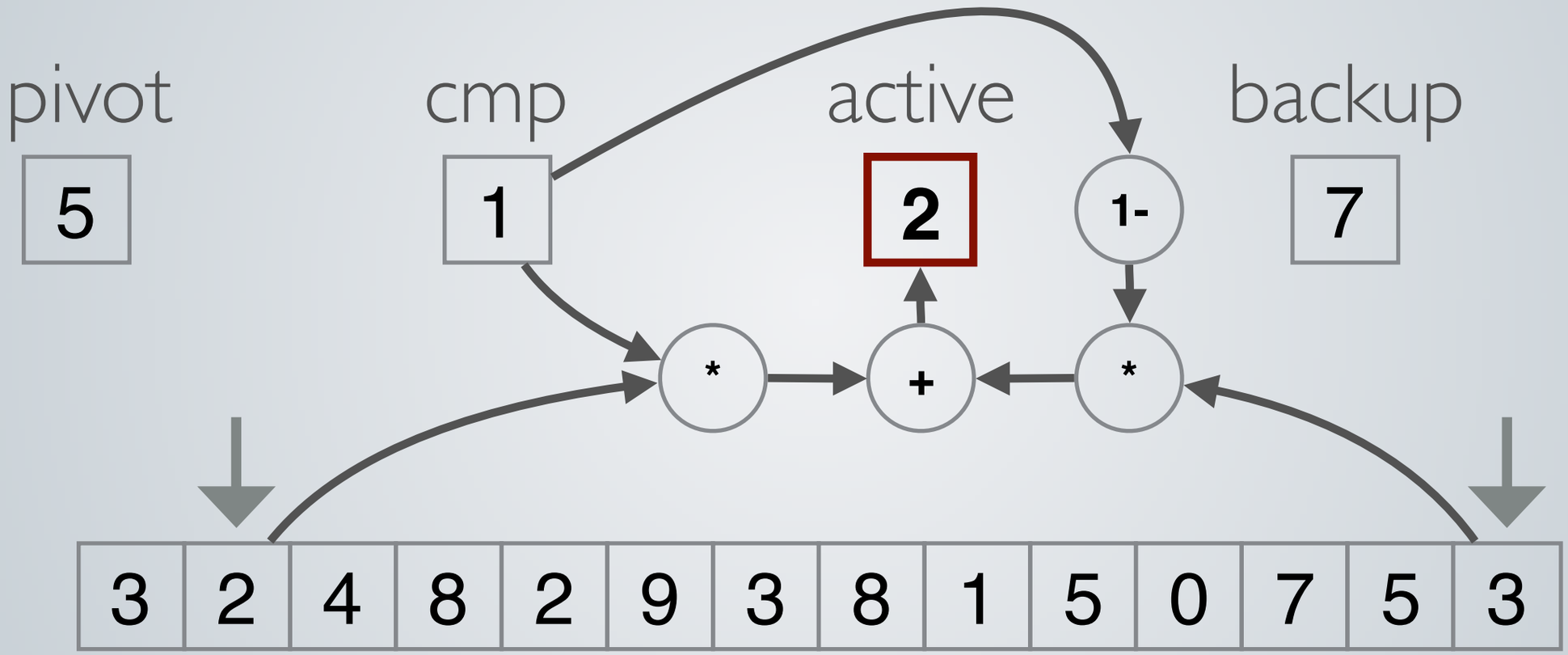
Evaluate Predicat & Write

PREDICATED CRACKING



Advance Cursor

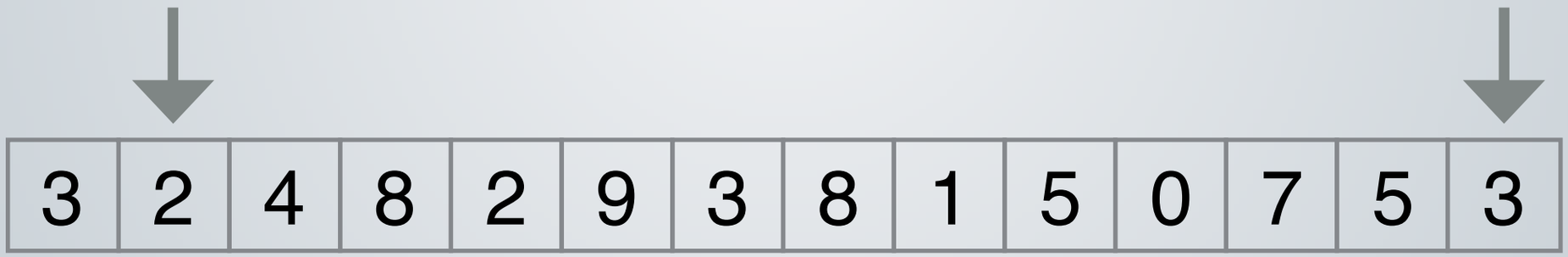
PREDICATED CRACKING



Read Next Element

PREDICATED CRACKING

pivot	cmp	backup	active
5	1	2	7



CWI

monetdb

PREDICATED CRACKING

- Predication for in-place algorithms
- No branching⇒No branch mispredictions
- Somewhat intricate
- Lots of copying stuff around (integer granularity ⇒ inefficient)
- Bulk-copying would be more efficient

VECTORIZED CRACKING

CWI

monetdb

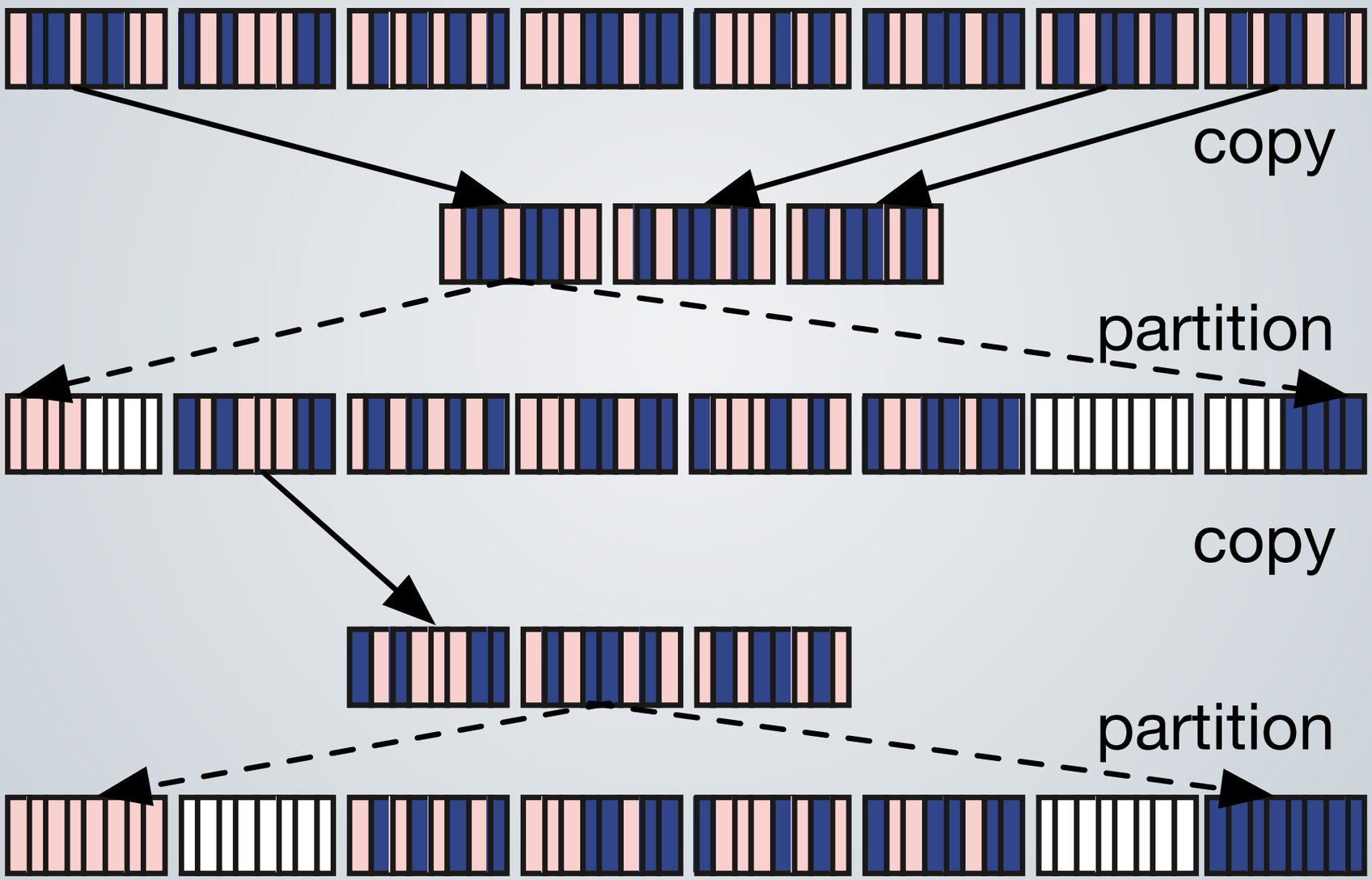
VECTORIZED CRACKING

- Turns in-place cracking into out-of-place cracking
- Copies Vector-sized chunks and cracks them into the array
  - Makes vanilla-predication possible
  - Uses SIMD-copying for vector copying
- Challenge: ensure that values aren't “accidentally" overwritten

CWI

monetdb

VECTORIZED CRACKING



CWI

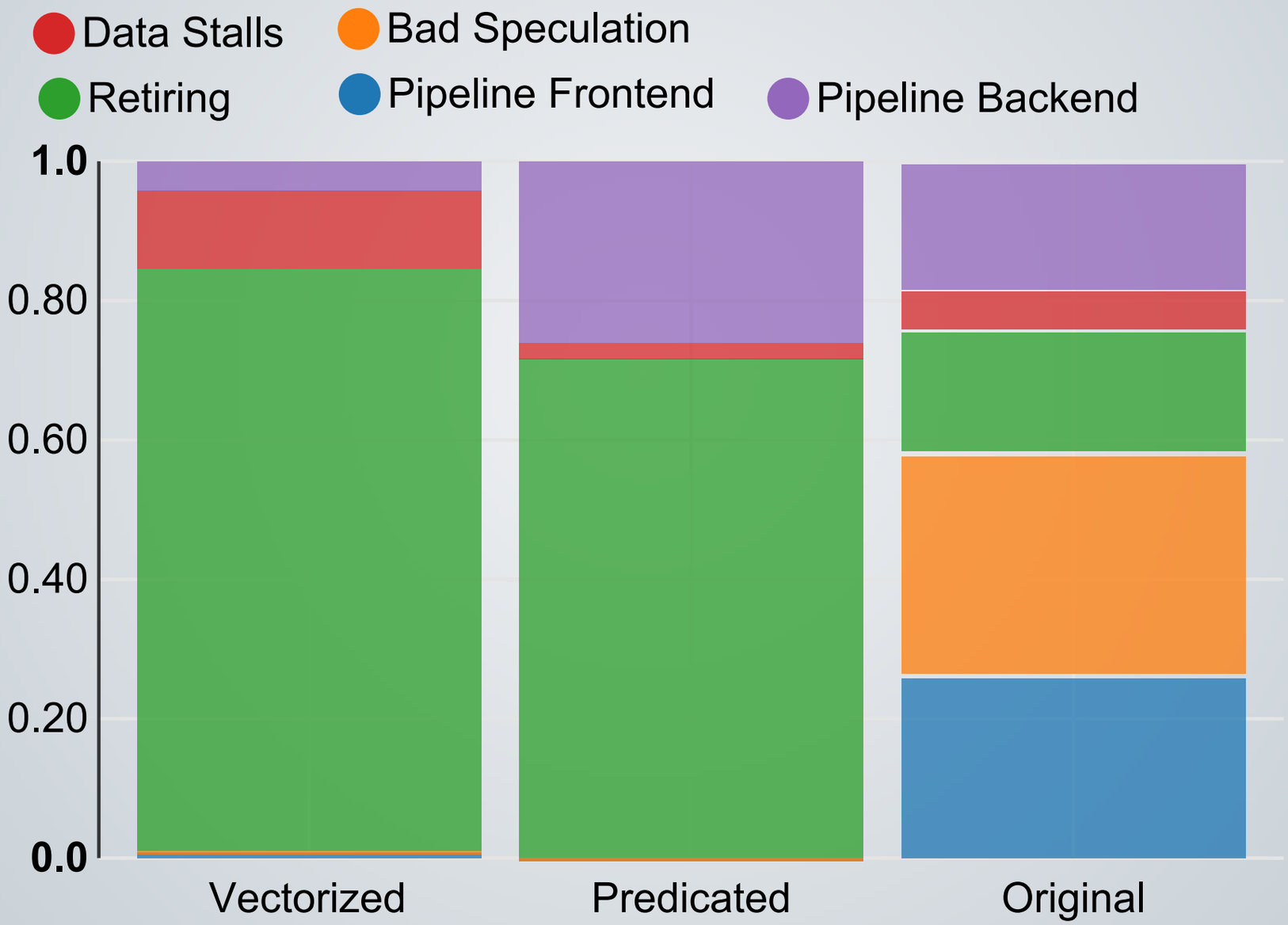
monetdb

RESULTS

CWI

monetdb

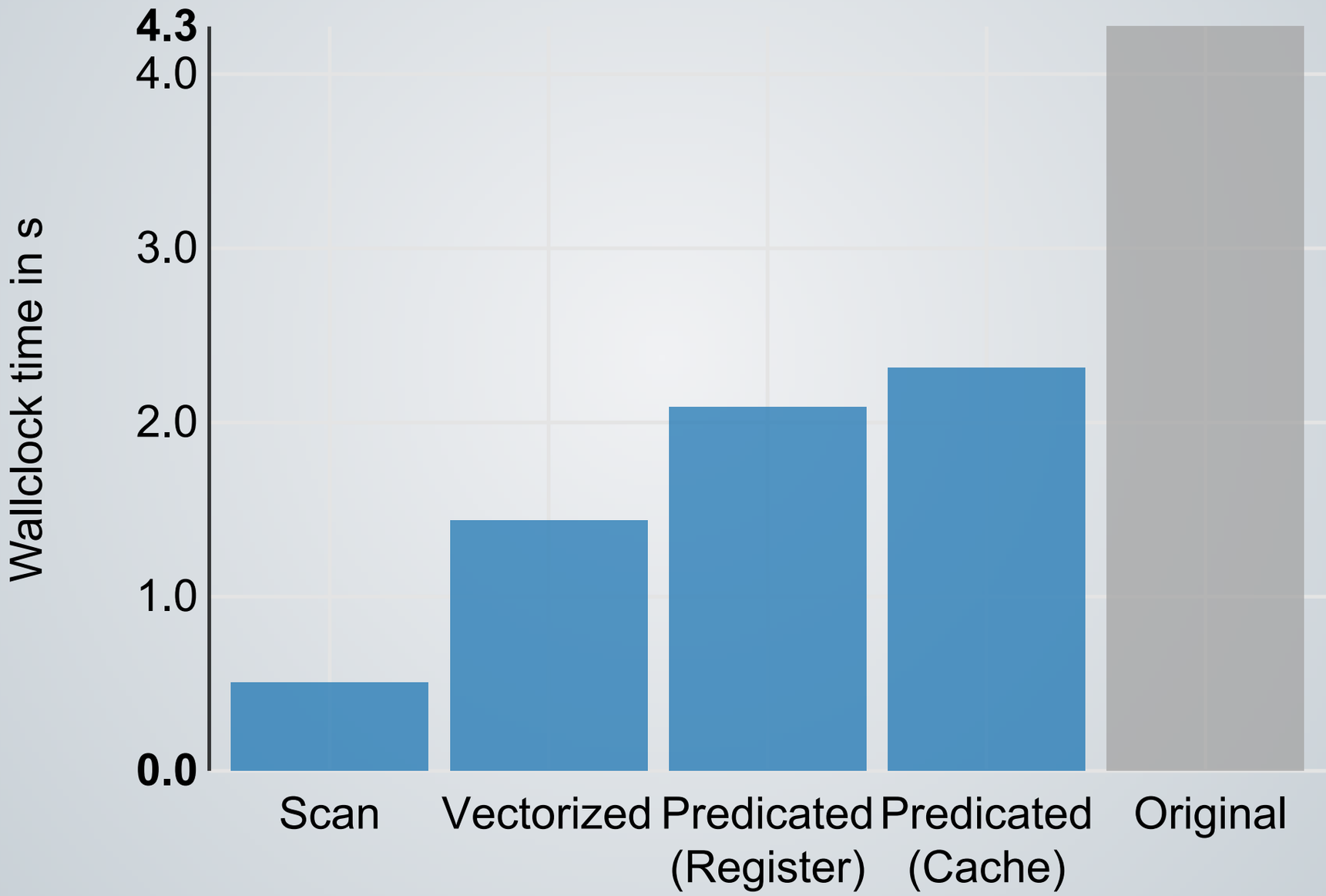
RESULTS



CWI

monetdb

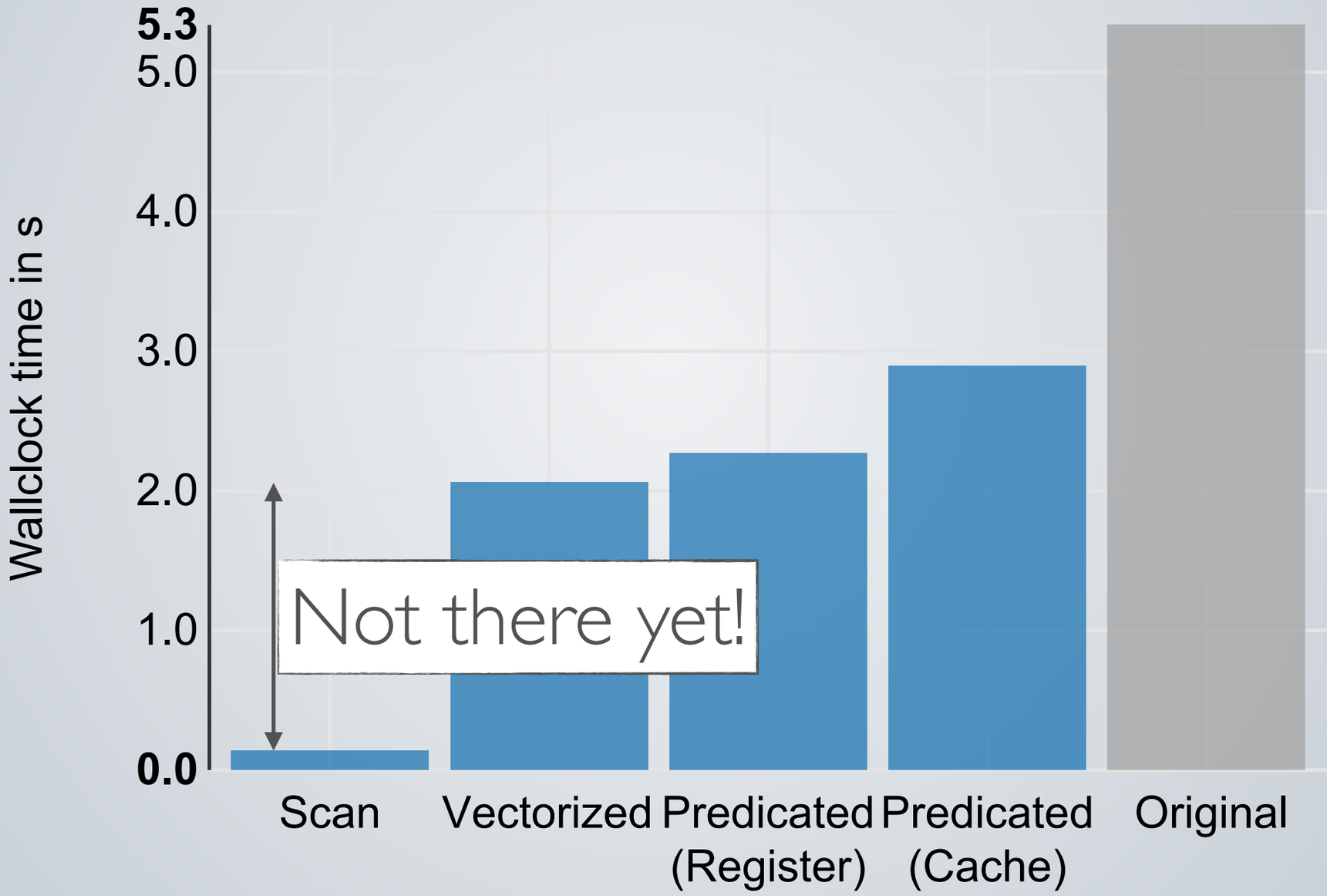
RESULTS: WORKSTATION



CWI

monetdb

RESULTS: SERVER



CWI

monetdb

PARALLELIZATION

CWI

monetdb

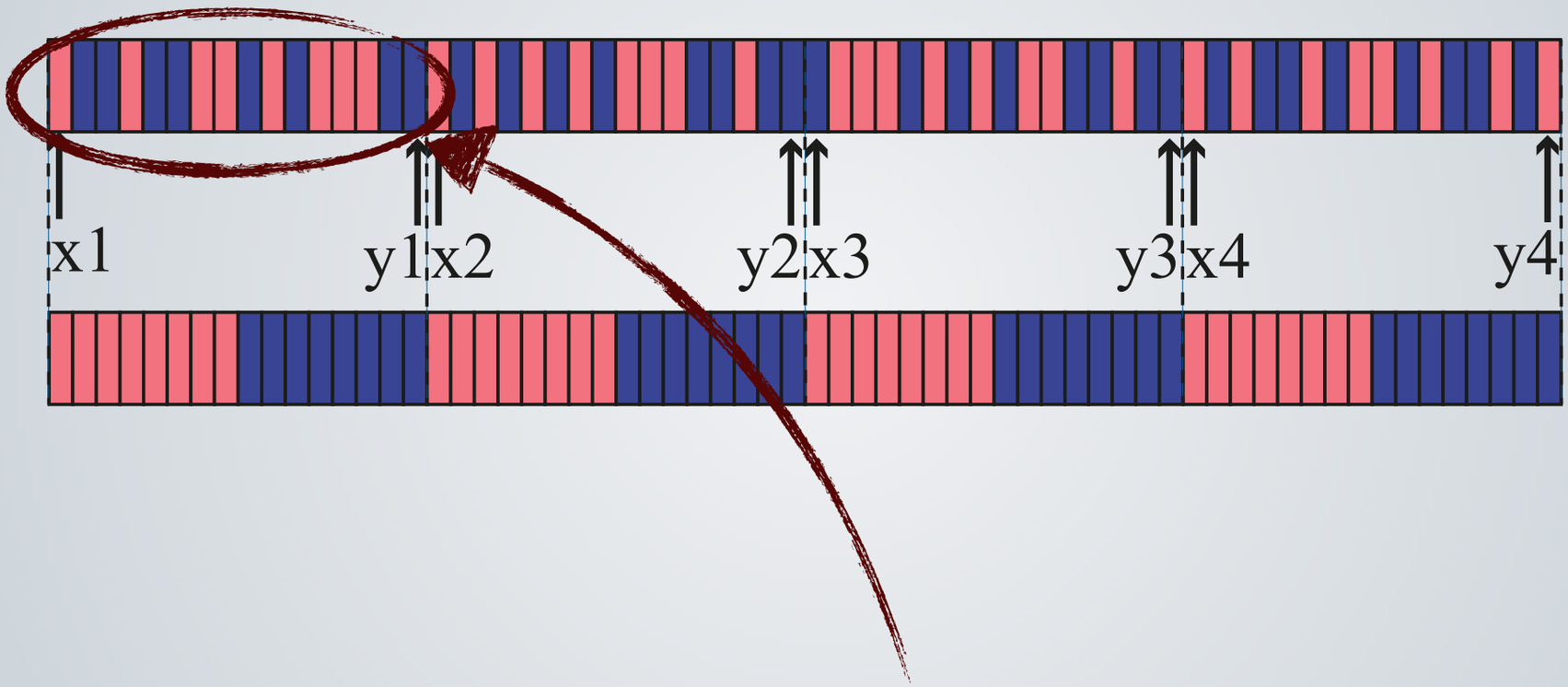
PARALLELIZATION

- Obvious Solution: Partitioning

CWI

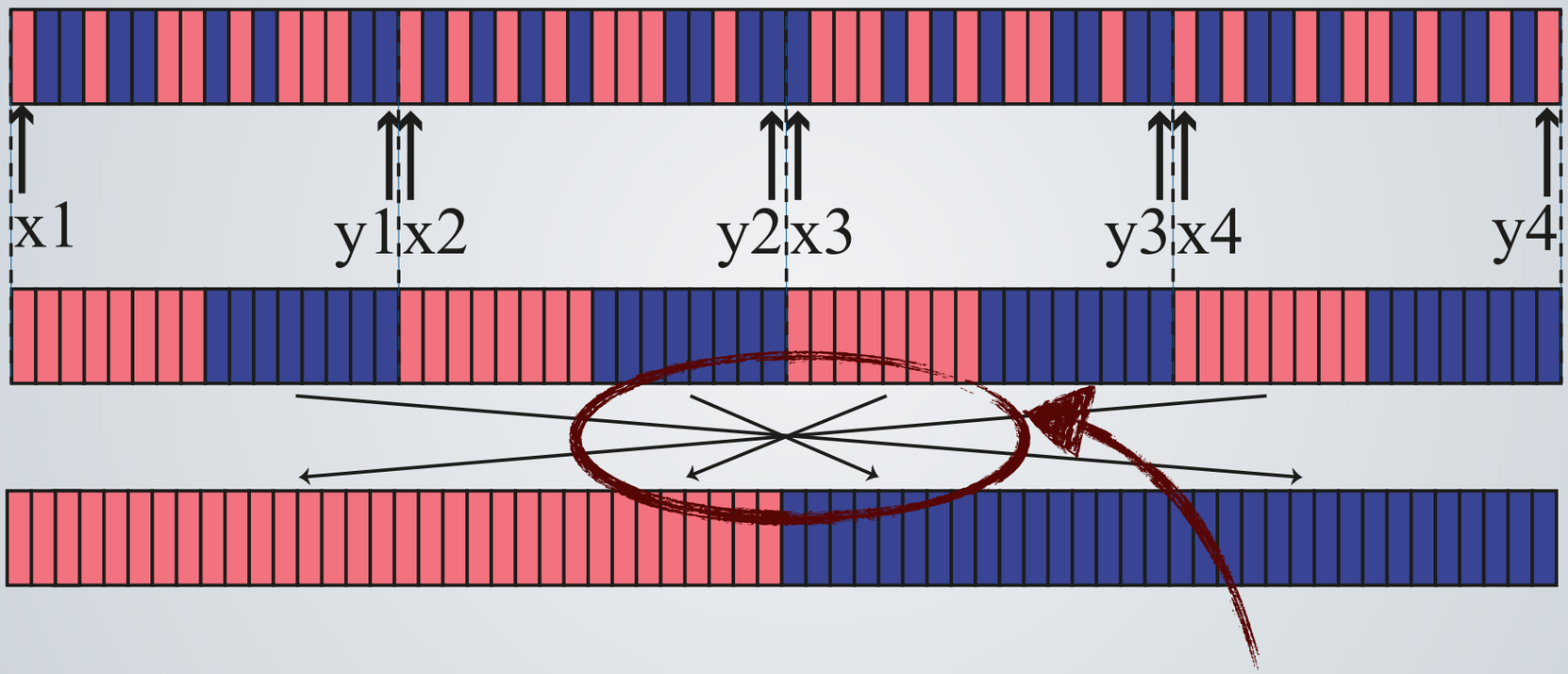
monetdb

CRACK & MERGE

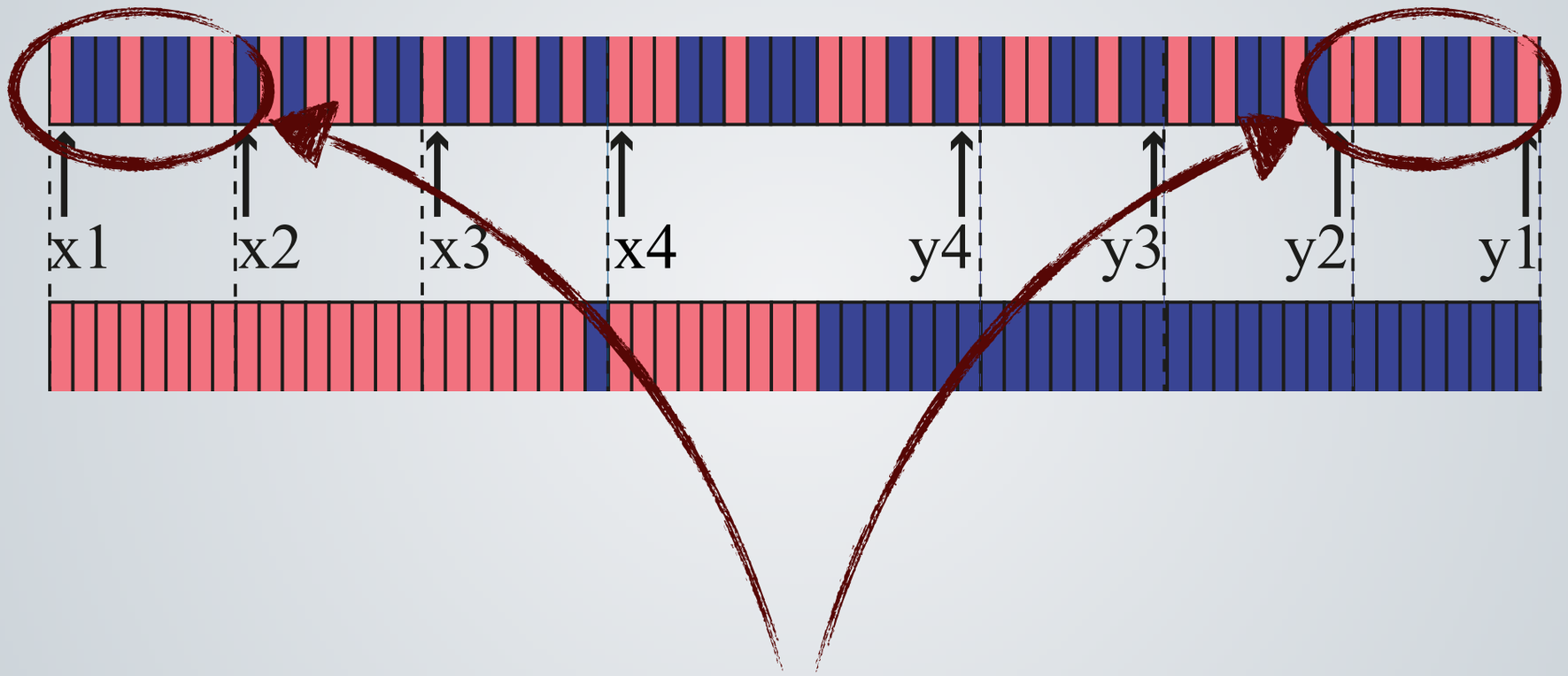


Partition

CRACK & MERGE



REFINED CRACK & MERGE

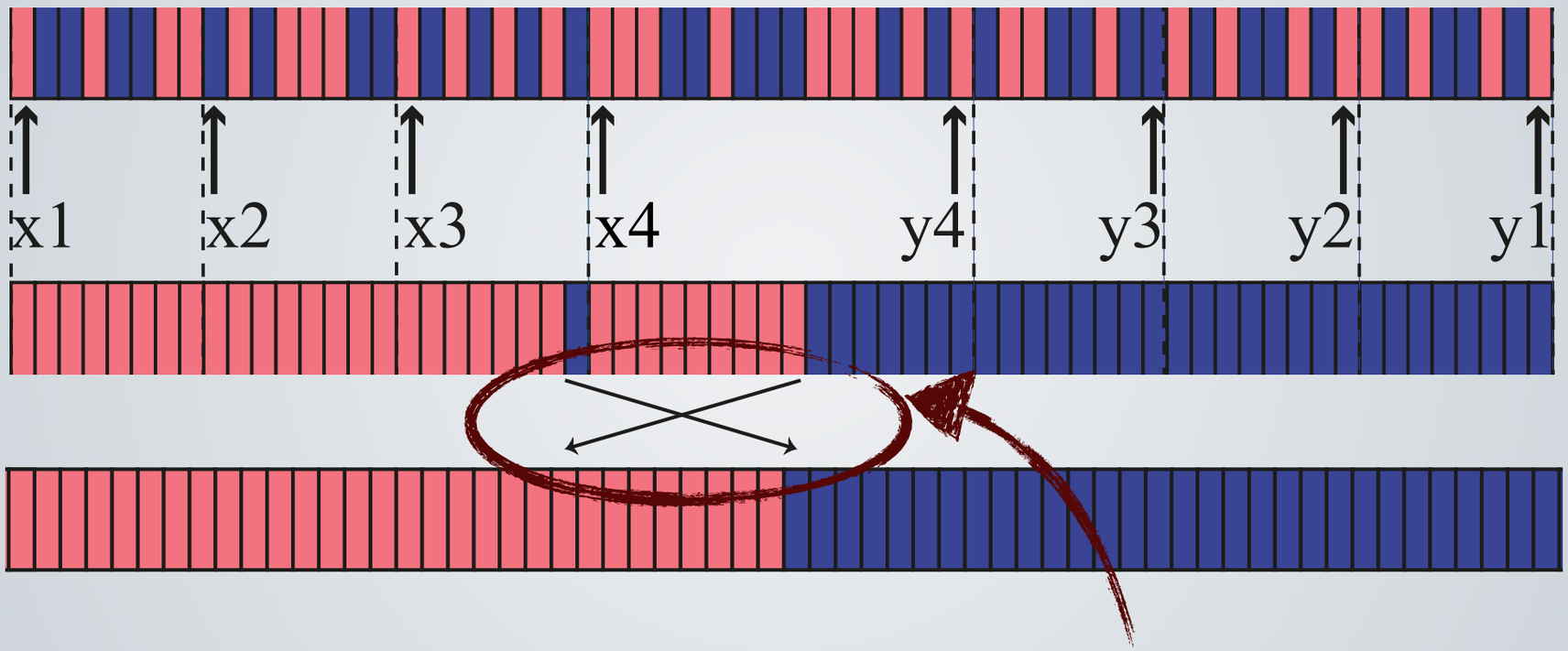


Partition

CWI

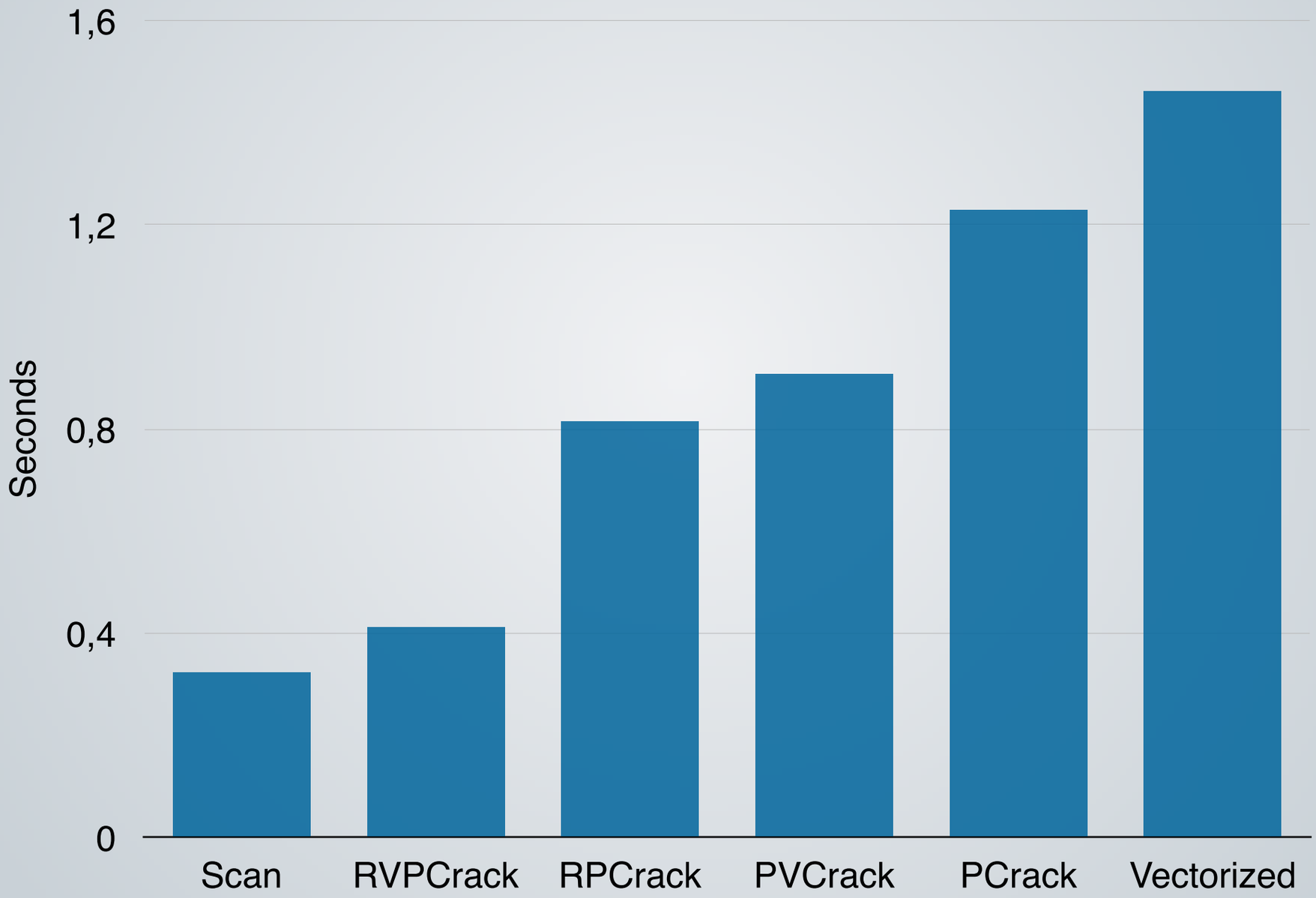
monetdb

REFINED CRACK & MERGE



Smaller Merge

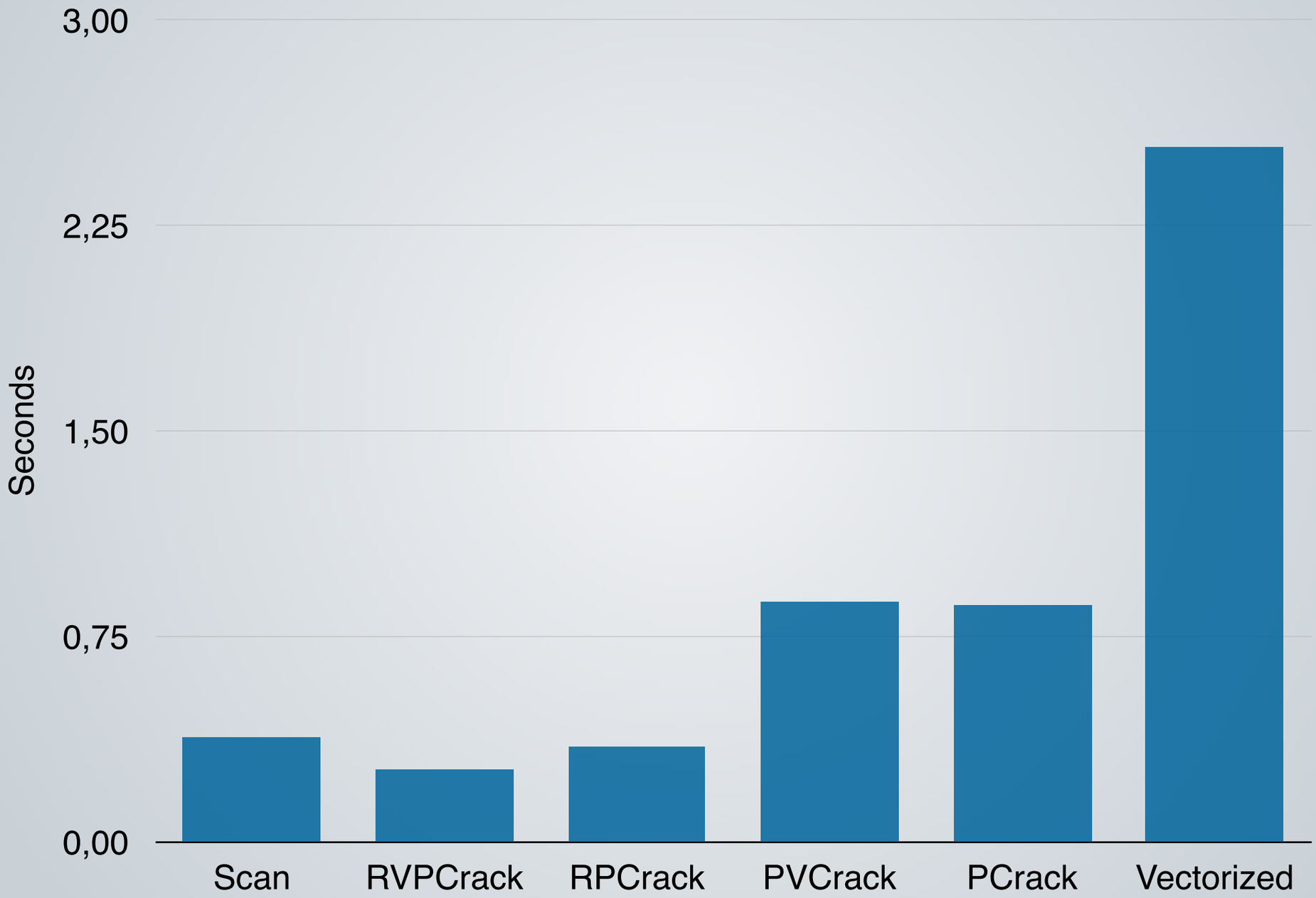
RESULTS: WORKSTATION



CWI

monetdb

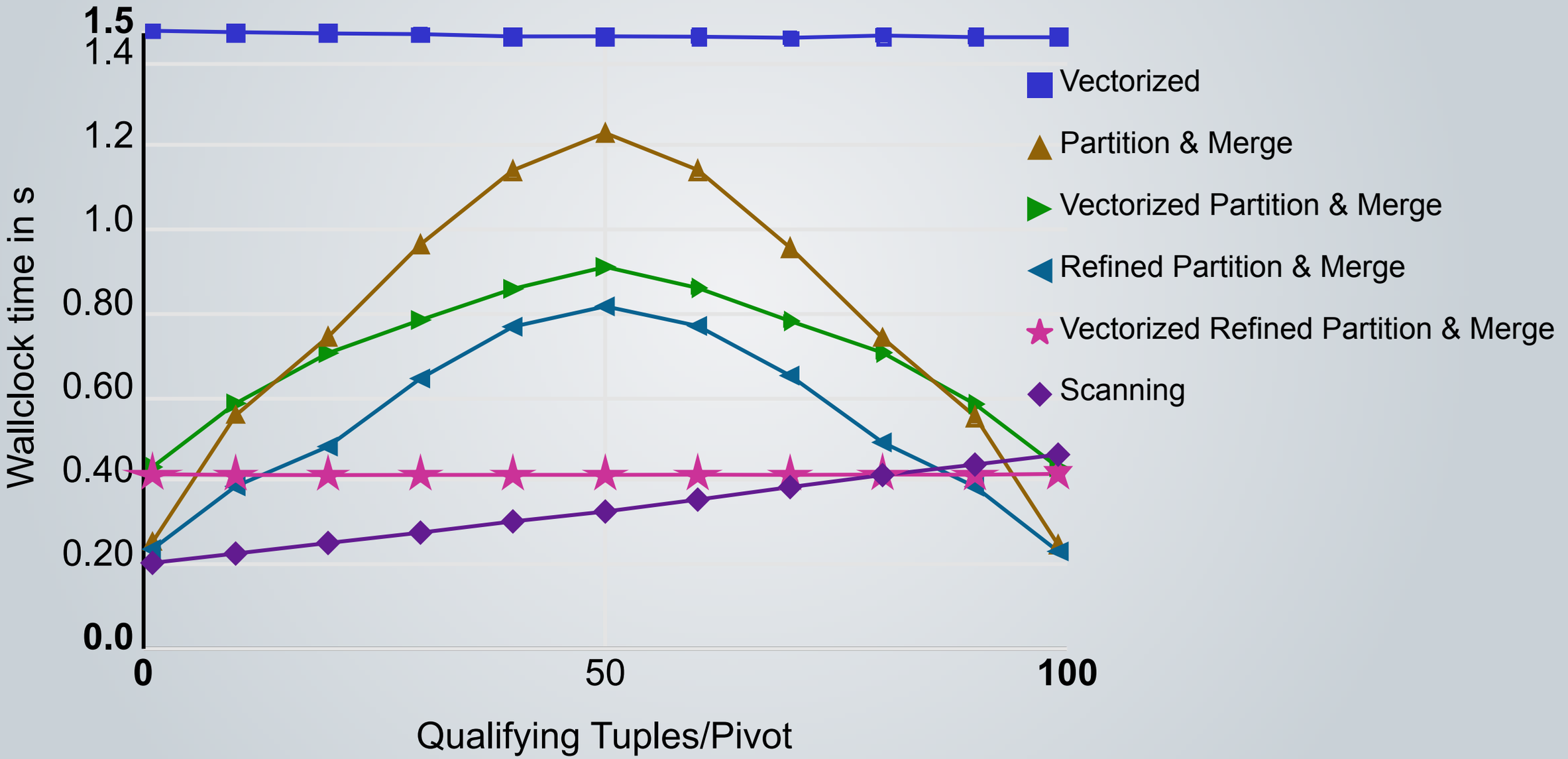
RESULTS: SERVER



CWI

monetdb

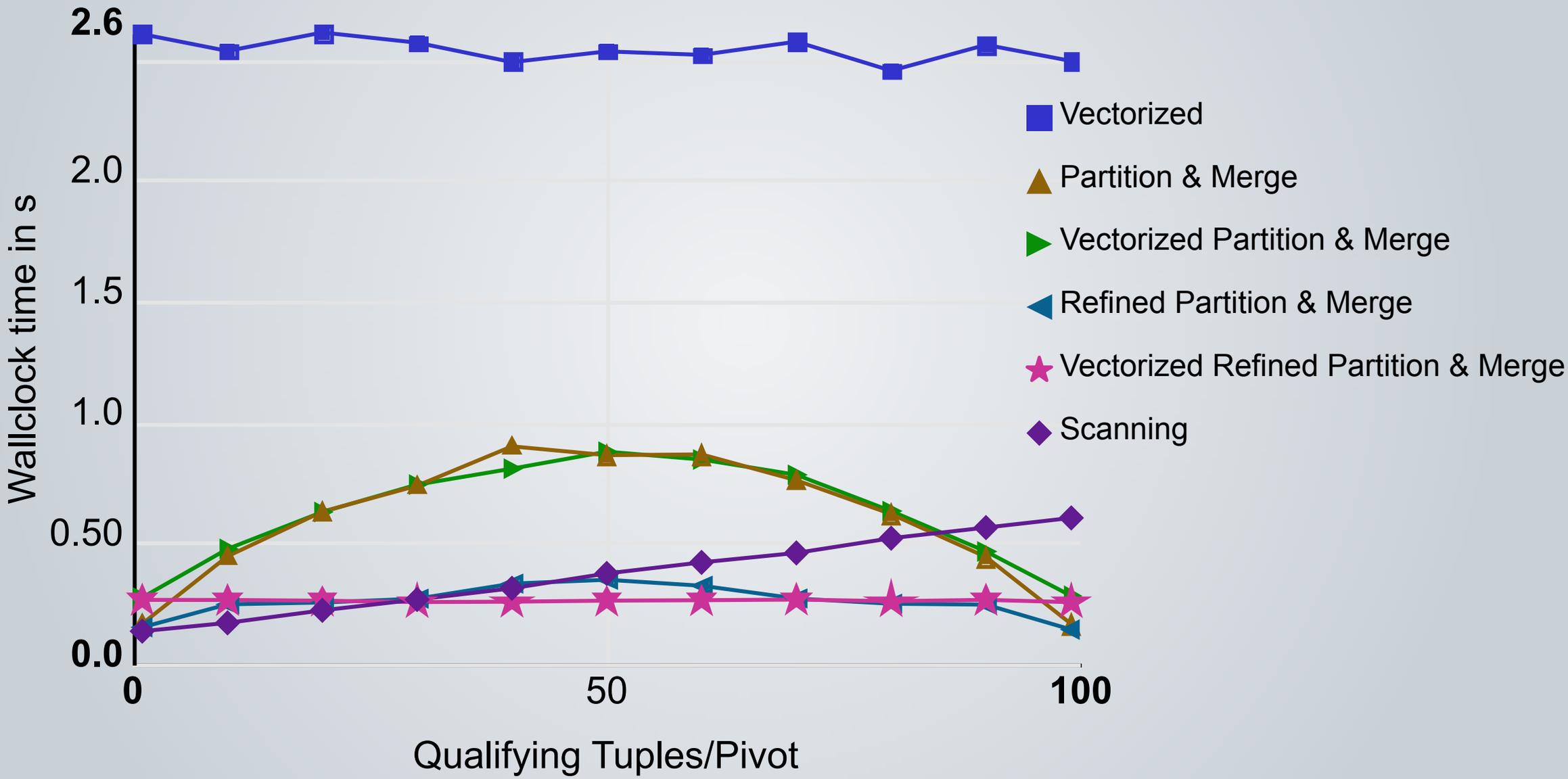
IMPACT OF SELECTIVITY: WORKSTATION



CWI

monetdb

IMPACT OF SELECTIVITY: SERVER



CWI

monetdb

CONCLUSIONS

CWI

monetdb