

F1NGERPR1NTS: PROCESSING JUST THE BITS YOU NEED, AND NONE OF THE 1s YOU DON'T

Roger Sayle and John Mayfield

NextMove Software, Cambridge, UK

Sheffield Chemoinformatics Conference, Sheffield, UK, Wednesday 21st June 2023



OVERVIEW

- Chemical Similarity and Tanimoto Coefficient
- The von Neumann Bottleneck
- Swamidass and Baldi BitBounds
- Arthor 3.x JIT
- GPU Memory Strides
- Bit Transposition/Inverted Index
- Bit Addition Tree



SET THEORY

- The Tanimoto Coefficient is one measure of similarity between a pair of sets.

$$\frac{|A \cap B|}{|A \cup B|}$$

- Axiom: $|A| + |B| = |A \cup B| + |A \cap B|$
- Hence $\frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} = \frac{|A| + |B| - |A \cup B|}{|A \cup B|}$
- If **A** and **B** are constant, Tanimoto (and others) can be calculated from just the intersect: **popcount(A&B)**

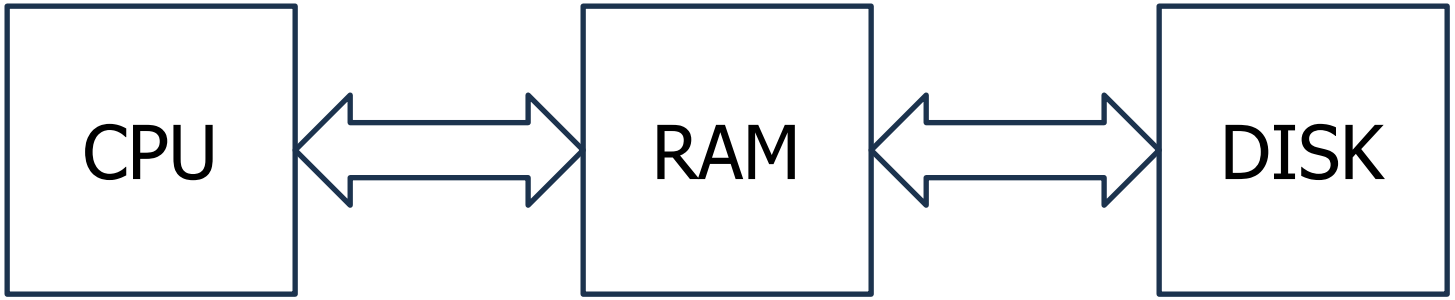


HOW BIG IS THE PROBLEM?

- For similarity searching, **1 billion** molecules requires:
  - **34GB** 256 bit fingerprint
  - **68GB** 512 bit fingerprint
  - **136GB** 1024 bit fingerprint
  - **252GB** 2048 bit fingerprint
  - **512GB** 4096 bit fingerprint
- **1TB** of memory can hold Enamine REAL 2023 (**6 billion**) as either 256, 512, or 1024 bit fingerprints.
- ZINC22 (**35 billion**) even 256-bit does not fit!



VON NEUMANN ARCHITECTURE



- All processing takes place in the CPU, and therefore is rate limited by the bandwidth between RAM and CPU, or for larger tasks from DISK.



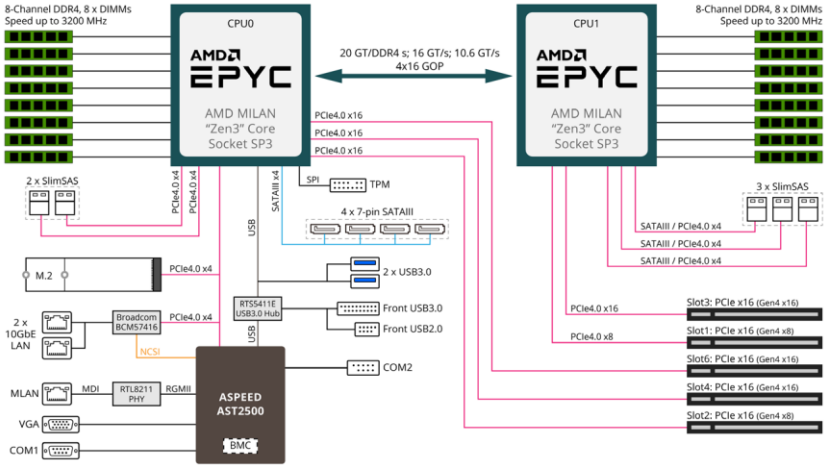
MEMORY BANDWIDTH

- A 256-bit fingerprint is 32 bytes.
- Memory bandwidth is MHz*bus width (8 bytes)
  - DDR3 1600 MHz 12.8 GB/s per channel, 0.42B 256-bit FP/s
  - DDR4 2133 MHz 15.9 GB/s per channel, 0.53B 256-bit FP/s
  - DDR4 3200 MHz 23.8 GB/s per channel, 0.79B 256-bits FP/s
  - DDR5 4800 MHz 35.7 GB/s per channel, 1.19B 256-bit FP/s
- Normally multiple channels of RAM per CPU socket
  - 2nd Gen Xeon 6 ch @ 2934 MHz per sock, 4.40B 256-bit FP/s
  - 3rd Gen EPYC 8 ch @ 3200 MHz per sock, 6.38B 256-bit FP/s
  - 4th Gen Xeon 8 ch @ 4800 MHz per sock, 9.5B 256-bit FP/s
  - 4th Gen EPYC 12 ch @ 4800 MHz per sock, 14.3B 256-bit FP/s



MODERN COMPUTERS



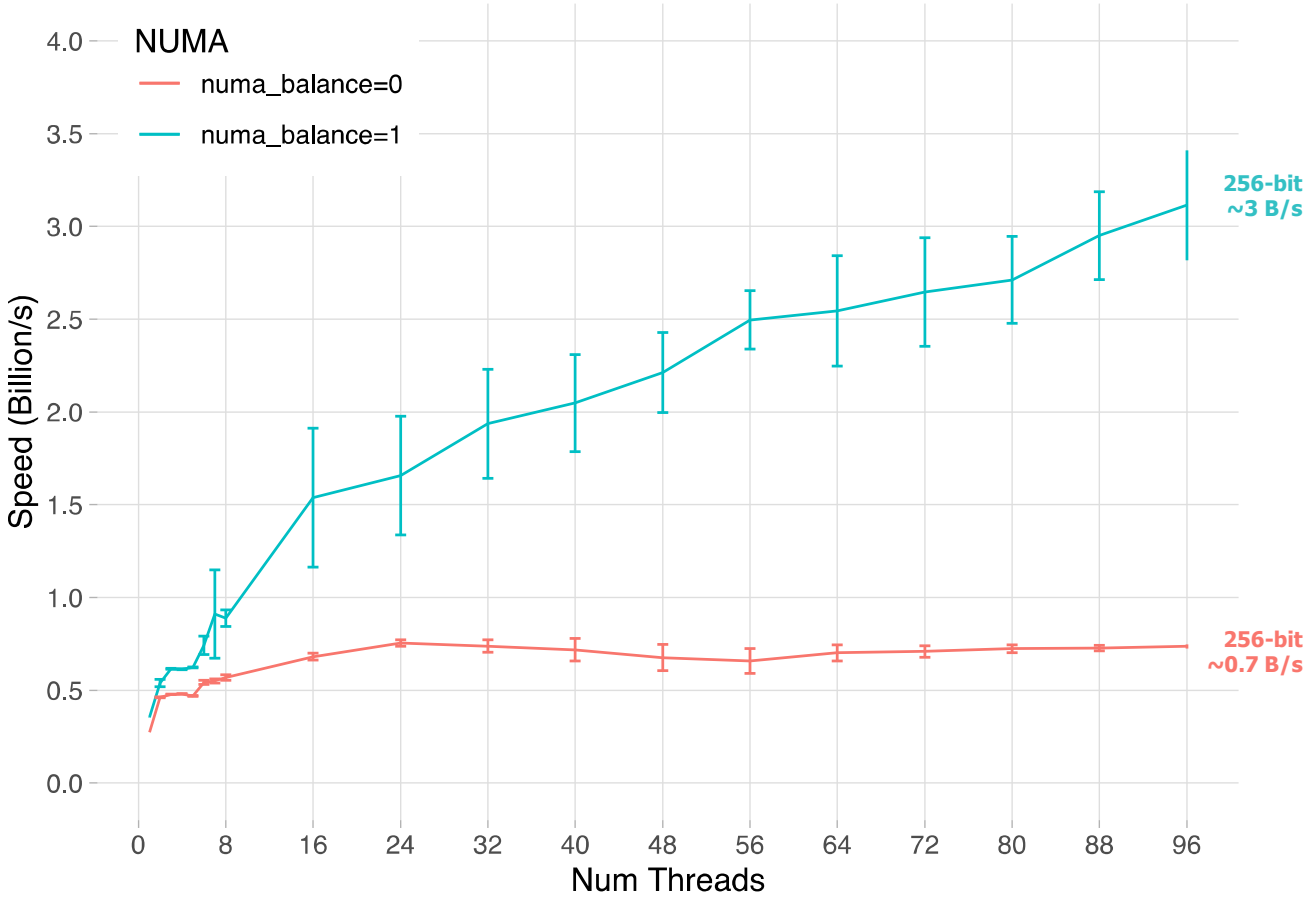


Gigabyte MZ72-HB0 Motherboard, Non-Uniform Memory Access (NUMA):

```
% numactl -H
available: 8 nodes (0-7)
node 0 cpus: 0 1 2 3 4 5 48 49 50 51 52 53
node 1 cpus: 6 7 8 9 10 11 54 55 56 57 58 59
node 2 cpus: 12 13 14 15 16 17 60 61 62 63 64 65
node 3 cpus: 18 19 20 21 22 23 66 67 68 69 70 71
node 4 cpus: 24 25 26 27 28 29 72 73 74 75 76 77
node 5 cpus: 30 31 32 33 34 35 78 79 80 81 82 83
node 6 cpus: 36 37 38 39 40 41 84 85 86 87 88 89
node 7 cpus: 42 43 44 45 46 47 90 91 92 93 94 95
node distances:
node   0   1   2   3   4   5   6   7
  0:  10  12  12  12  32  32  32  32
  1:  12  10  12  12  32  32  32  32
  2:  12  12  10  12  32  32  32  32
  3:  12  12  12  10  32  32  32  32
  4:  32  32  32  32  10  12  12  12
  5:  32  32  32  32  12  10  12  12
  6:  32  32  32  32  12  12  10  12
  7:  32  32  32  32  12  12  12  10
```



NUMA EFFECTS



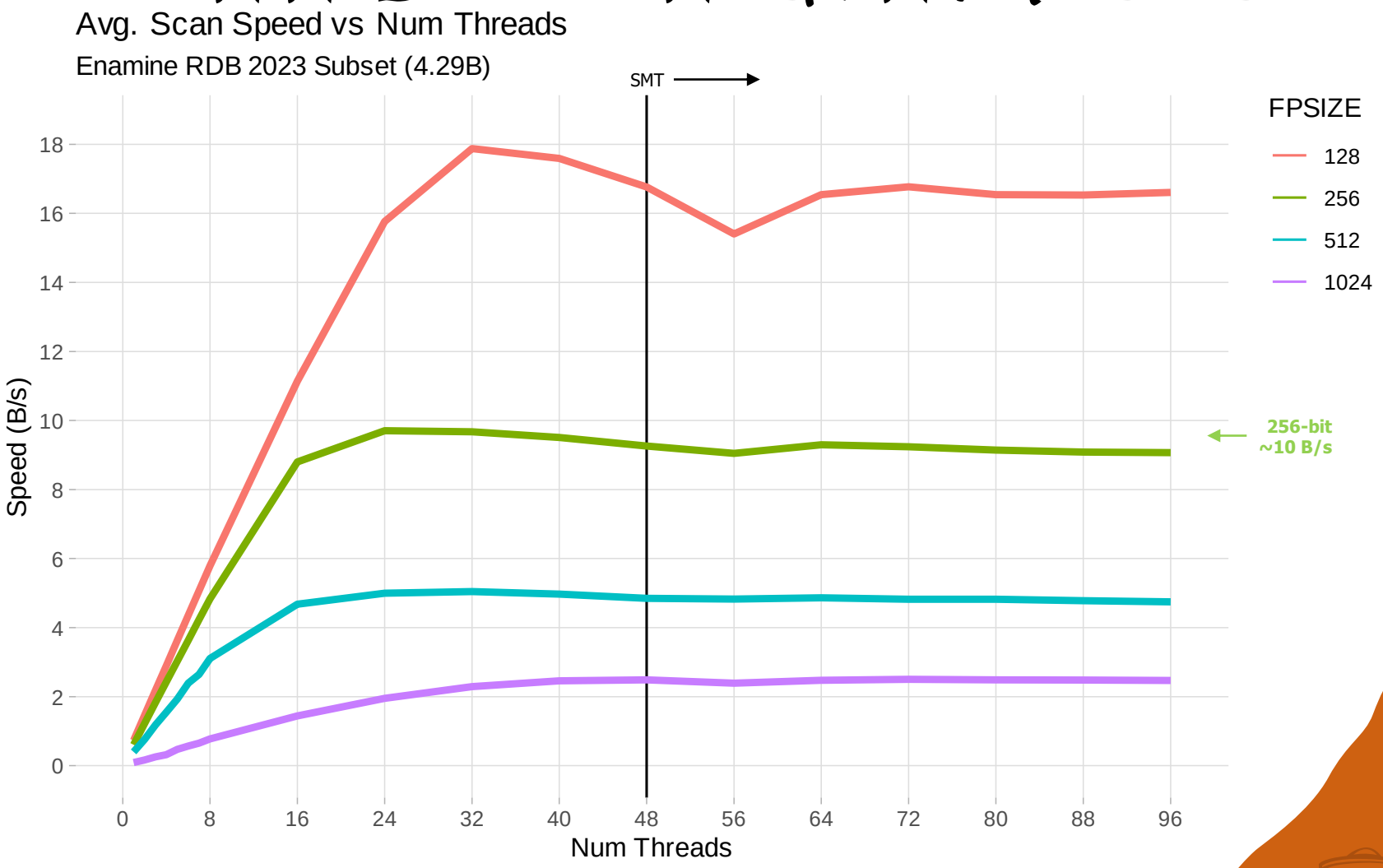
FPSim2 (FOSS) supports parallel queries and uses hardware instructions

Modern CPU/RAM is fast, the POP hardware instruction saturates 2ch DDR3, now need multiple threads

NUMA complications but operating system does well with auto balancing

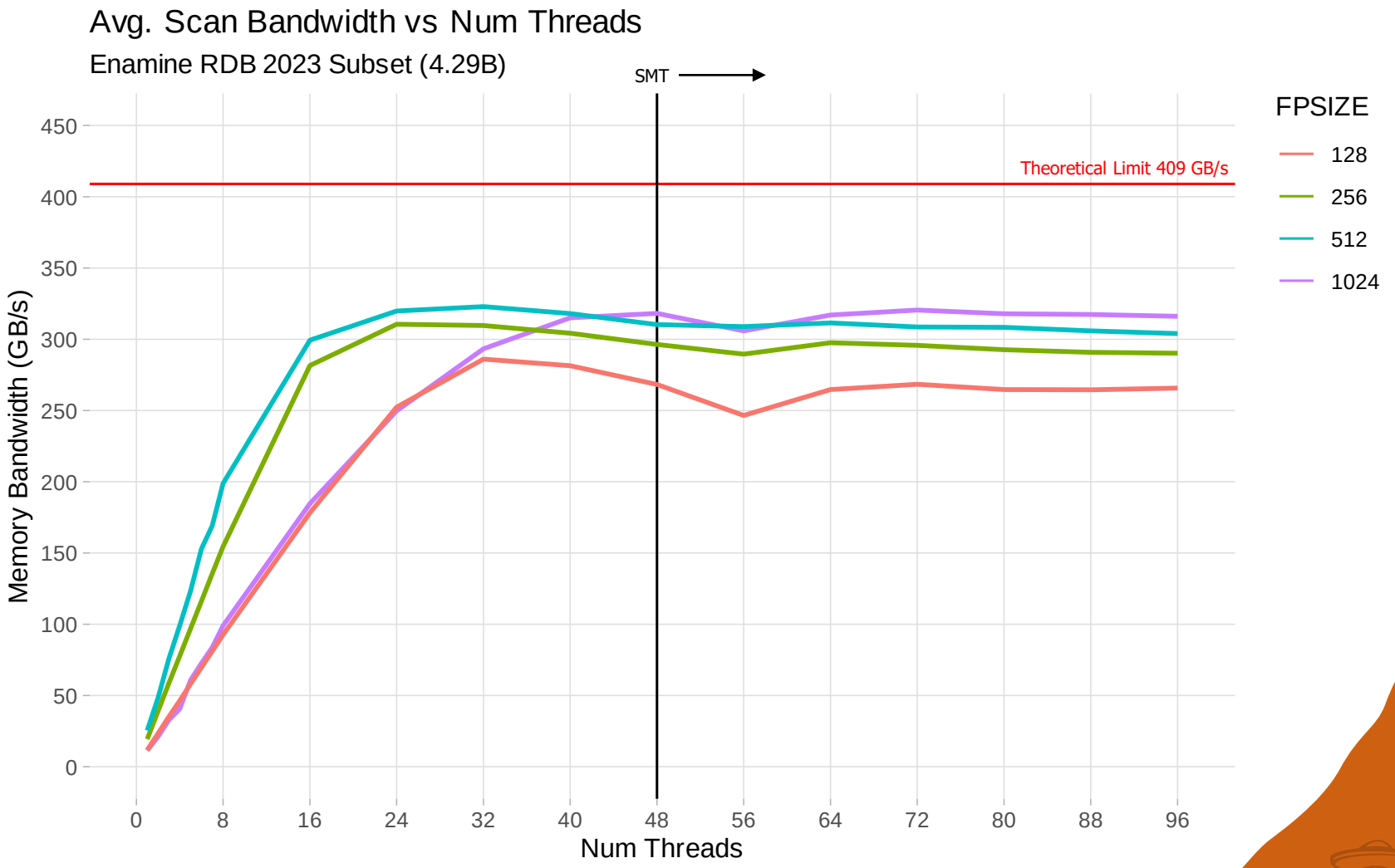
Benchmark: 19 top selling small molecules vs Enamine 2023, 256-bit ECFP4
Hardware: 2x AMD EPYC 7443, 1TB DDDR4 @ 3200 MHz
FPSim2: https://github.com/chembl/FPSim2

VON N. BOTTLENECK IN PRACTICE



Benchmark: 19 top selling small molecules vs Enamine 2023, 256-bit ECFP4
Hardware: 2x AMD EPYC 7443, 1TB DDDR4 @ 3200 MHz

VON N. BOTTLENECK IN PRACTICE



Benchmark: 19 top selling small molecules vs Enamine 2023, 256-bit ECFP4
Hardware: 2x AMD EPYC 7443, 1TB DDDR4 @ 3200 MHz

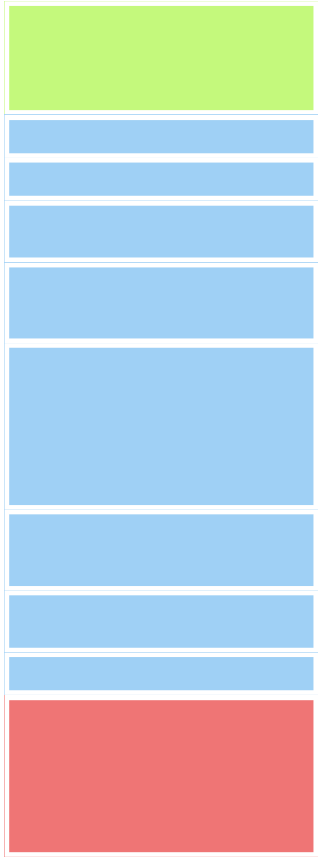
IN A NUTSHELL

- *"To go faster we need to look at less data"*
  - John Mayfield

...or use more hardware



SWAMIDASS & BALDI BIT BOUNDS



Partition the index into buckets based on POPC (number of on bits)

For a query **A** bits and bucket with **B** bits

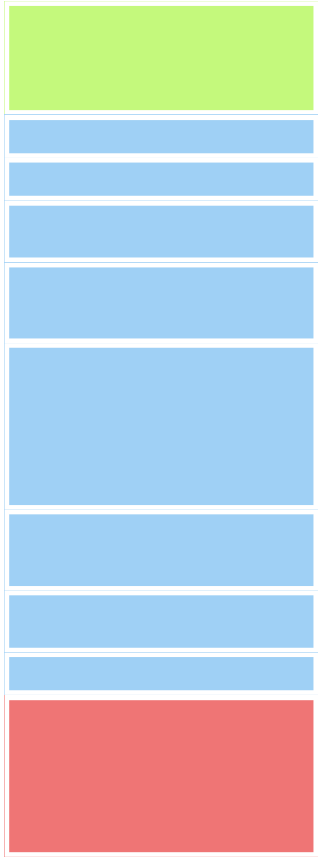
```
max_common = min(A,B)
t_max = max_common / (A+B-max_common)
```

Don't need to check the bucket if the best score (**t_max**) is less than required:

- Poor memory access pattern
- NUMA unfriendly
- Less useful for sparse FP (ECFP/Morgan)

Bounds and Algorithms for Fast Exact Searches of Chemical Fingerprints in Linear and Sublinear Time
S. Joshua Swamidass and Pierre Baldi. *Journal of Chemical Information and Modeling* **2007** *47* (2), 302-317
DOI: 10.1021/ci600358f

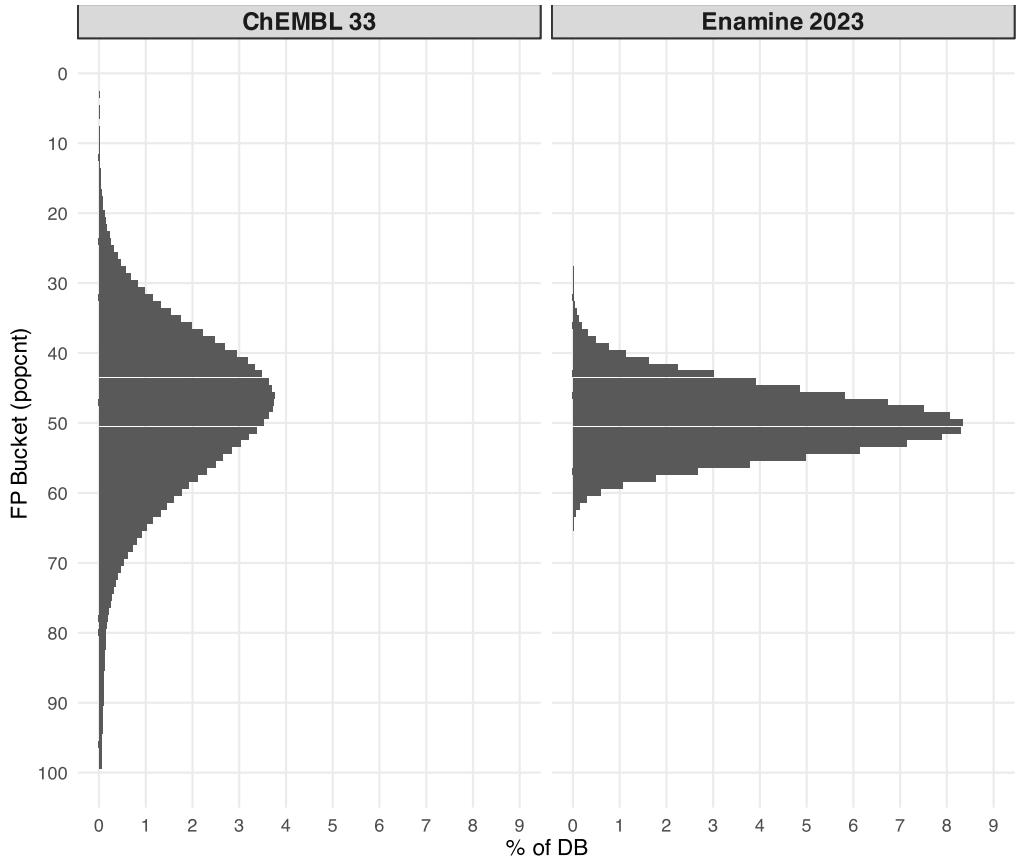
SWAMIDASS & BALDI BIT BOUNDS



Header

Fingerprint Index (buckets)

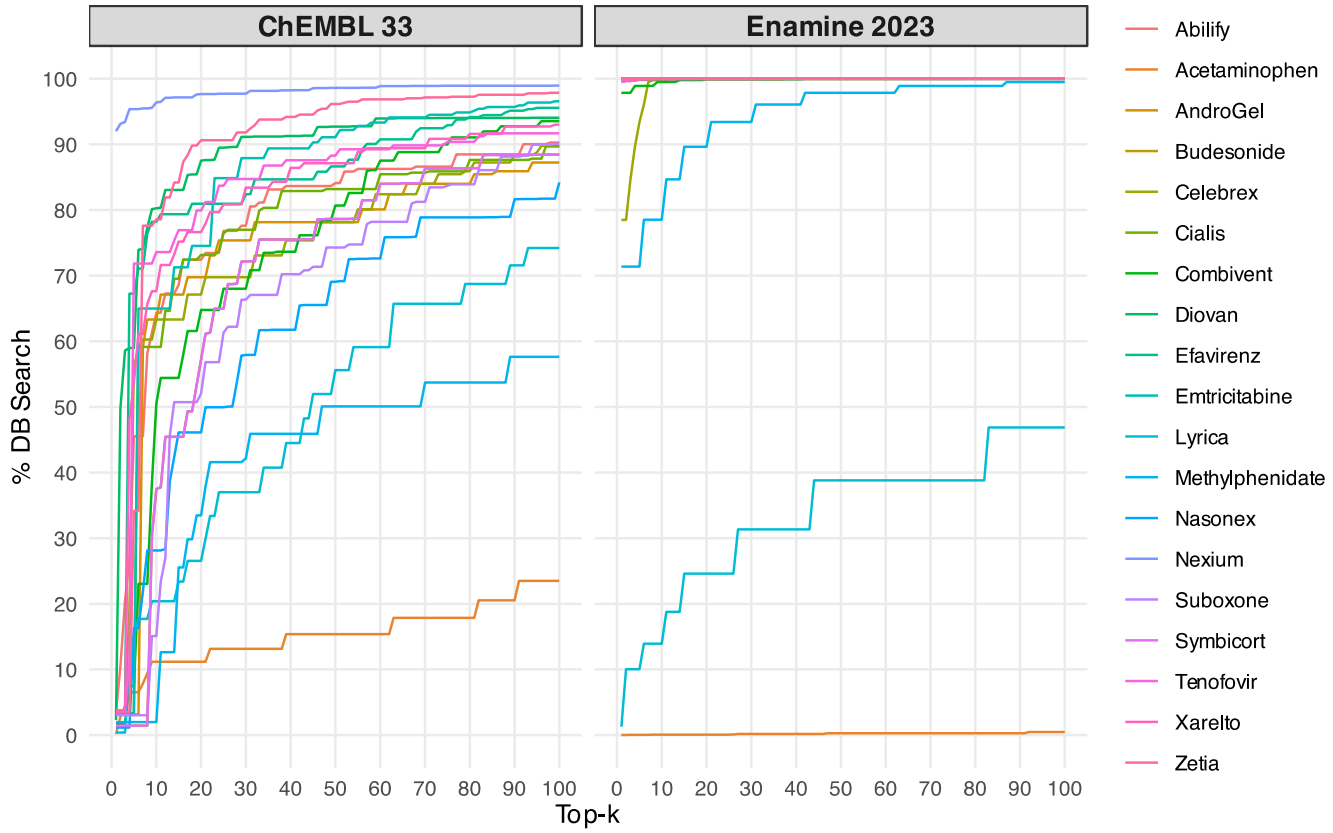
Data Index



Distribution of POPC depends on FP and database

Bounds and Algorithms for Fast Exact Searches of Chemical Fingerprints in Linear and Sublinear Time
S. Joshua Swamidass and Pierre Baldi. *Journal of Chemical Information and Modeling* **2007** *47* (2), 302-317
DOI: 10.1021/ci600358f

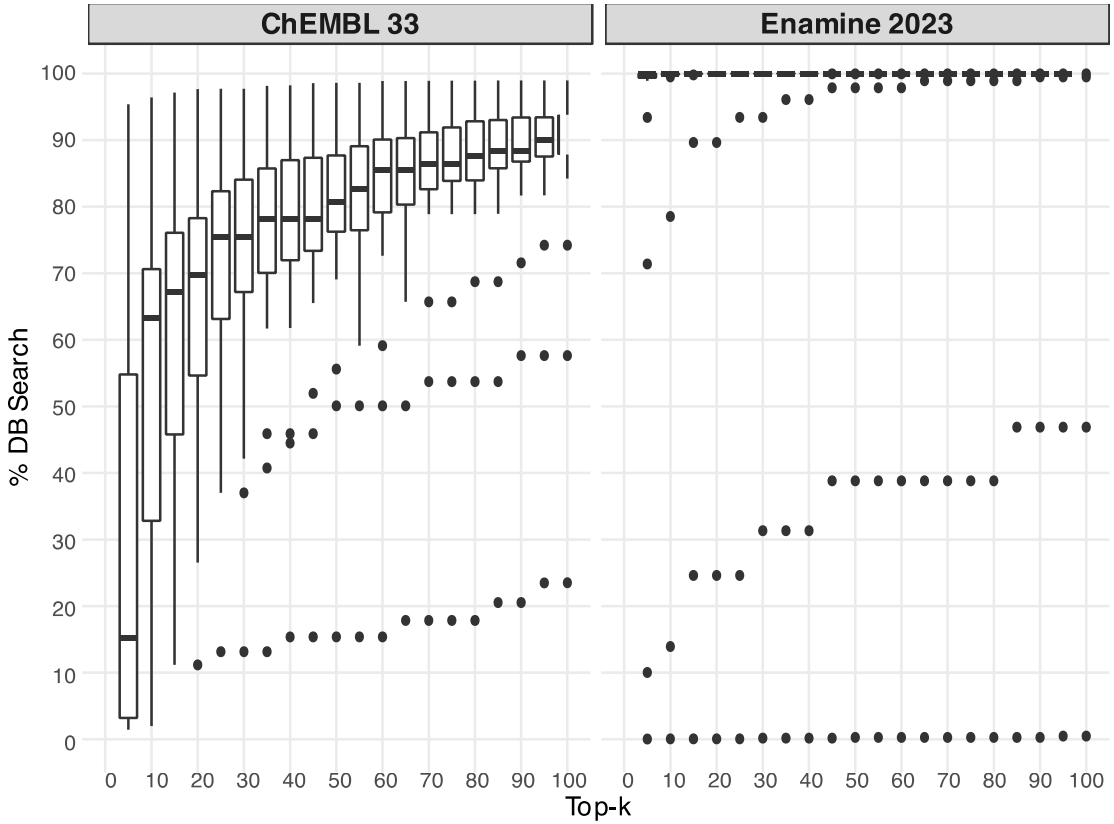
SWAMIDASS & BALDI BIT BOUNDS



Per-query: Except for exceptional queries (outliers) and the very top hits, we still need to look at most of the database. Most queries against Enamine we pretty much always end up looking at 100%



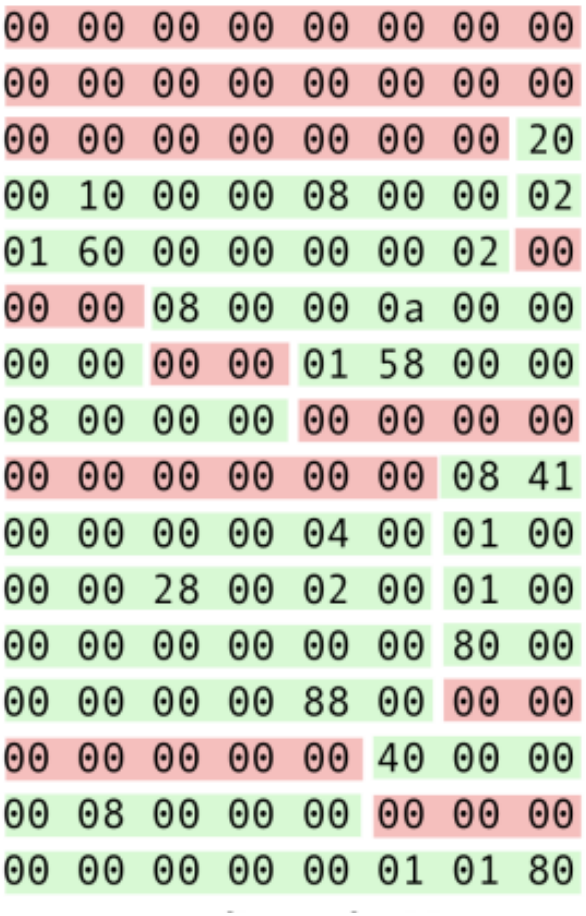
SWAMIDASS & BALDI BIT BOUNDS



Except for exceptional queries (outliers) and the very top hits, we still need to look at most of the database. Most queries against Enamine we pretty much always end up looking at 100%



ARTHOR 1.0 INSIGHT



Only need to **test** the 1's set in the query to get the intersect, skip 0's

- *cache/memory paging means they still get read*

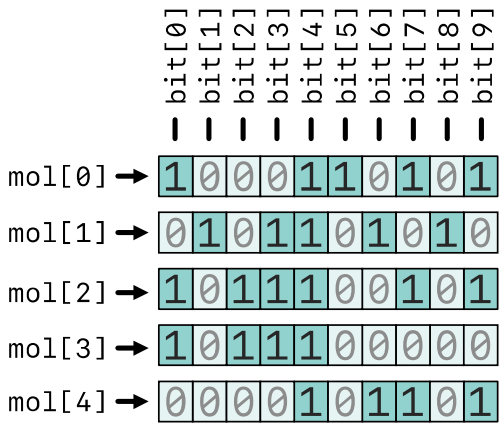
Consider a 1024-bit query with a **single bit**:

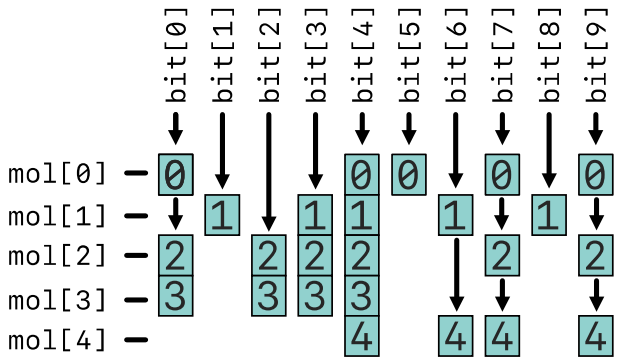
- 16x 64-bit POPC instructions **or**
- 1x BT (bit-test) instruction

Implemented as x86-64/ARM **Just-In-Time (JIT)** compiler



ARTHOR 4.0 INSIGHT





An *inverted index:*

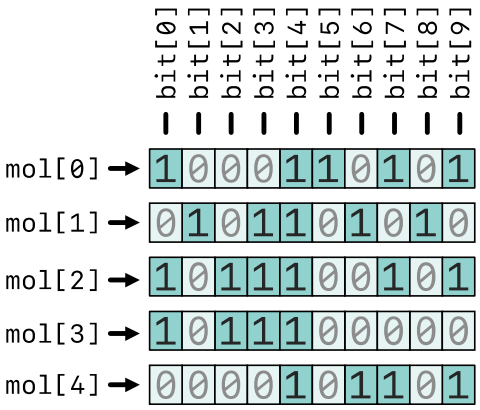
Rather than have the ***bits for each molecule*** we can have the ***molecules for each bit***

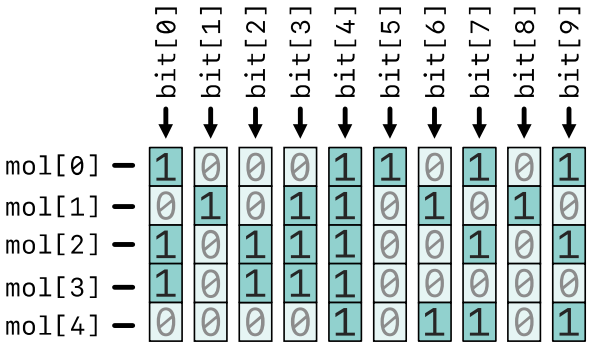
Compression techniques balance space vs decoding speed:

- list of integers
- list of gaps (varbyte)
- bitmap



ARTHOR 4.0 INSIGHT





Storing as a bitmap takes identical space (the transpose).

However now we only ever access the bits we need to check (those set in the query).

A query with ½ the bits (ρ0.5) in theory does ½ the memory accesses or I/O



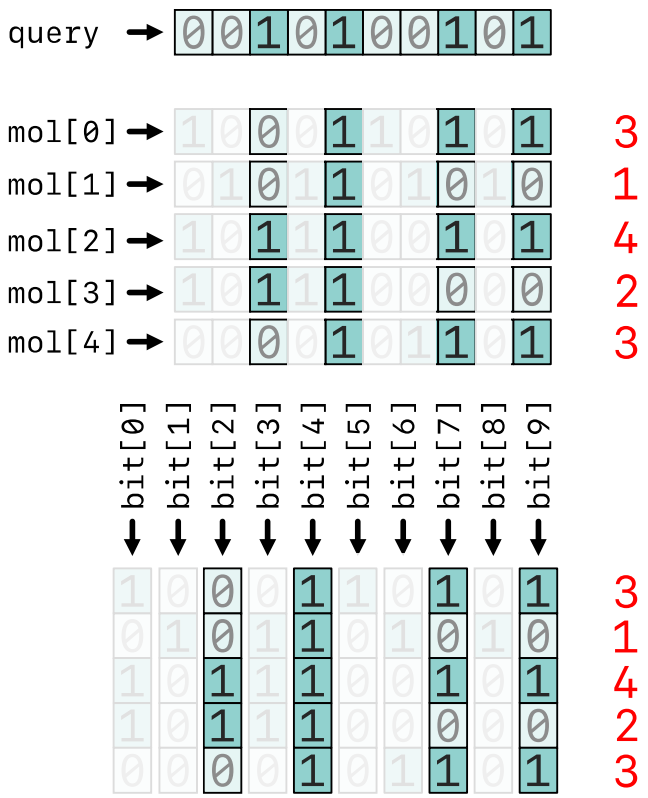
HOW MANY BITS IN A QUERY?

Query	256 Bits		1024 Bits		2048 Bits	
	POPC	%	POPC	%	POPC	%
Abilify	50	19.53	53	5.18	55	2.69
Acetaminophen	22	8.59	24	2.34	24	1.17
Androgel	41	16.02	43	4.20	43	2.10
Budesonide	62	24.22	67	6.54	67	3.27

- The number of bits is the query does not scale linearly with FP size
- A (small) fraction of the database



ARTHOR 4.0 INSIGHT



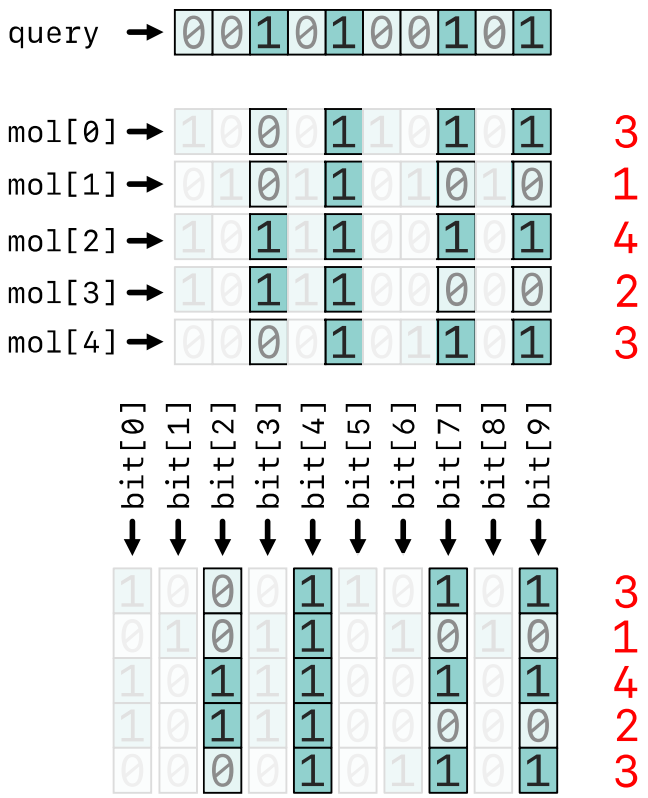
Conventional wisdom:

- **Intersect** / **Union** / **Not** operations
- Non-sequential mem. accesses
- Only v. large/sparse fingerprints
- Use posting-list compression
- Too much overhead

It couldn't possibly be useful on 256-bit FP?



ARTHOR 4.0 INSIGHT



Intersect / **Union** are efficiently implemented on bitmaps (&/|)

Only tells us if **all** features contain a record or **any** has the record.

Need an efficient way to **sum** the number of times a record occurs



THE TECHNICAL BIT

Code 1a. Basic **32-bit** implementation

```
unsigned char common[] = {0};

for (unsigned int bitmap : bitmaps) {
  for (unsigned int w=0; w<len; w++) {
    unsigned int word = bitmap[w];
    for (unsigned int bit : GetOnBits(word)) {
       common[32*w+bit]++;
    }
  }
}
```

Naïvely we could just turn the bitmap into a ***list of integers*** and accumulate how many times we see each one.

Code 1b. Better **32-bit** unrolled implementation

```
for (unsigned int bitmap : bitmaps) {
  for (unsigned int w=0; w<len; w++) {
    unsigned int word = bitmap[w];
    common[32*w]    += ((word&0x00000001)!=0);
    common[32*w+1]  += ((word&0x00000002)!=0);
    common[32*w+2]  += ((word&0x00000004)!=0);
    // ...
    common[32*w+30] += ((word&0x40000000)!=0);
    common[32*w+31] += ((word&0x80000000)!=0);
  }
}
```

Unrolled/branchless is faster



THE REALLY TECHNICAL BIT

Code 2. **32-bit** SIMD implementation

```
for (unsigned int bitmap : bitmaps) {
  unsigned int *dst = common;
  for (unsigned int w=0; i<len; i++) {
    unsigned int word = bitmap[w];
    *dst++ = word & 0x01010101;      // 0,8,16,24
    *dst++ = (word>>1) & 0x01010101; // 1,9,17,25
    *dst++ = (word>>2) & 0x01010101; // 2,10,18,26
    *dst++ = (word>>3) & 0x01010101; // 3,11,19,27
    *dst++ = (word>>4) & 0x01010101; // 4,12,20,28
    *dst++ = (word>>5) & 0x01010101; // 5,13,21,29
    *dst++ = (word>>6) & 0x01010101; // 6,14,22,30
    *dst++ = (word>>7) & 0x01010101; // 7,15,23,31
  }
}
```

Single instruction, multiple data (**SIMD**) technique can add **4x** 8-bits at once in a **32-bit** register.

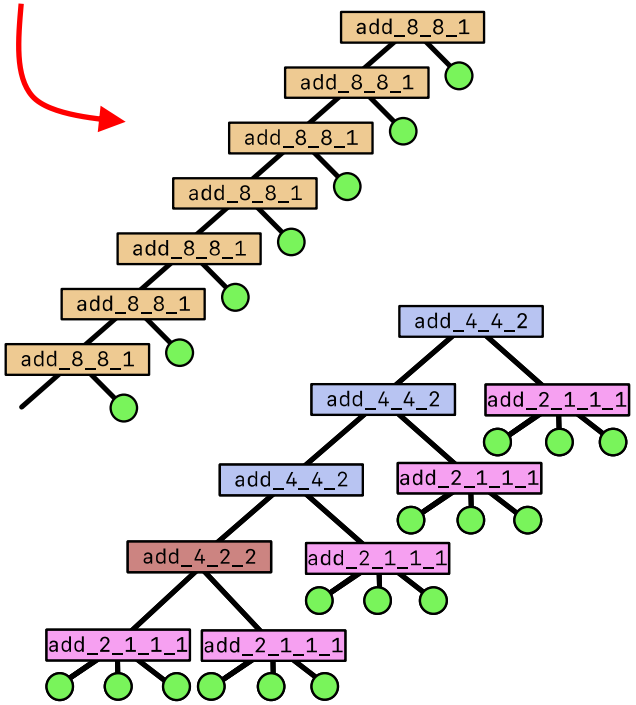
Easy to use 128, 256, and even 512-bit registers

Permute the storage order such that the results are *unshuffled*



THE REALLY REALLY TECHNICAL BIT

this was code 2



Difference size accumulators run at different speeds:

- 2-bit: ≤ 3 bits (16x @32-bit)
- 4-bit: ≤ 15 bits (8x @32-bit)
- 8-bit: ≤ 255 bits (4x @32-bit)

Addition tree uses less instructions, more efficient

1-bit input (inverted index)

add_8_8_1 8-bit out 8-bit and 1-bit in

add_4_4_2 4-bit out 4-bit and 2-bit in

add_4_2_2 4-bit out 2x 2-bit in

add_2_1_1_1 2-bit out 3x 1-bit in



DOES IT WORK?



EXPECTATIONS

- Performance will depend on Query POPC
- Less memory scanned but more more work to do
  - more efficient when a bottle necked (i.e. high parallelization)
- POPC/density inflection point at which is less efficient

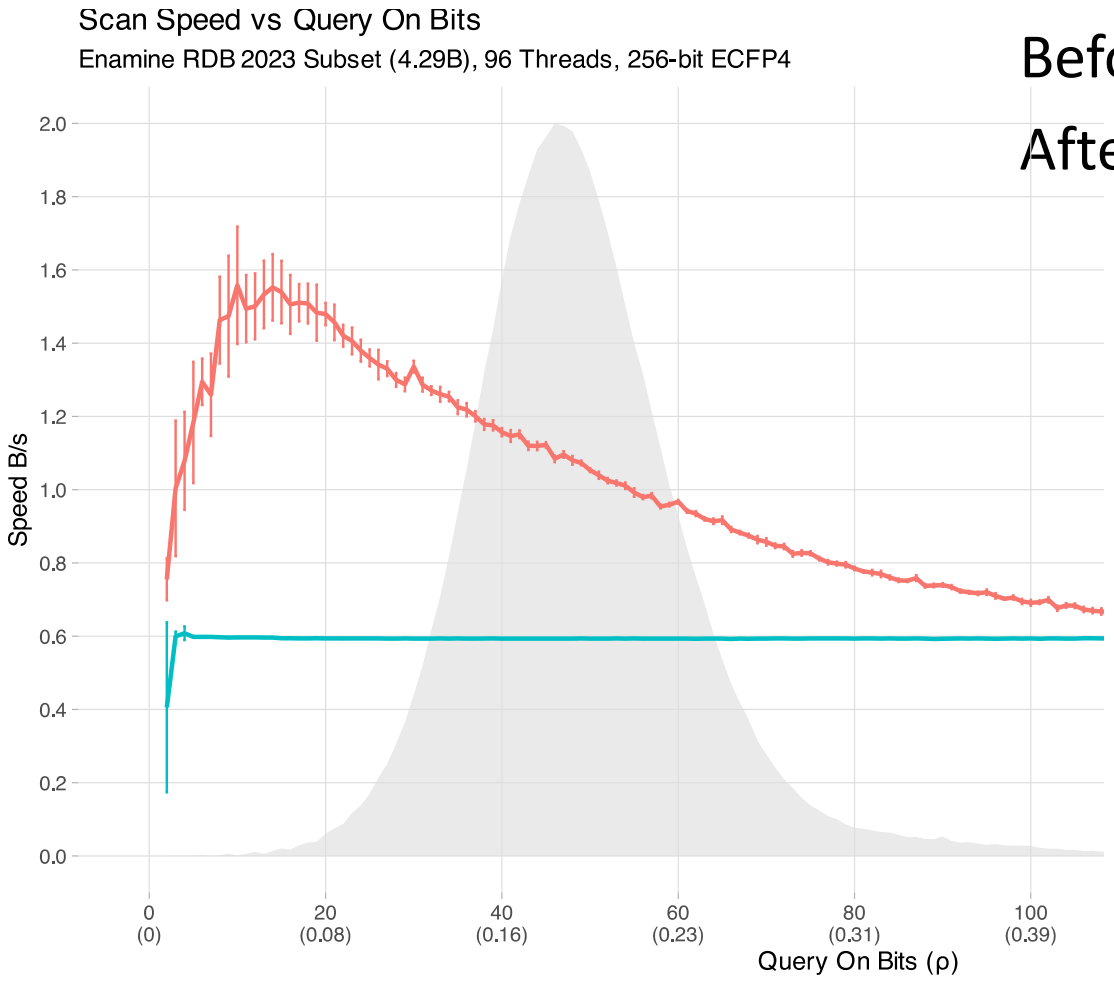
Queries: Random sample from CHEMBL of different POPCs

Targets: Enamine REAL 2023 (4.29B*)



* We store up to 4.29 billion (32-bit addressing) per-file part, 6 billion is two parts

SINGLE THREAD REALITY

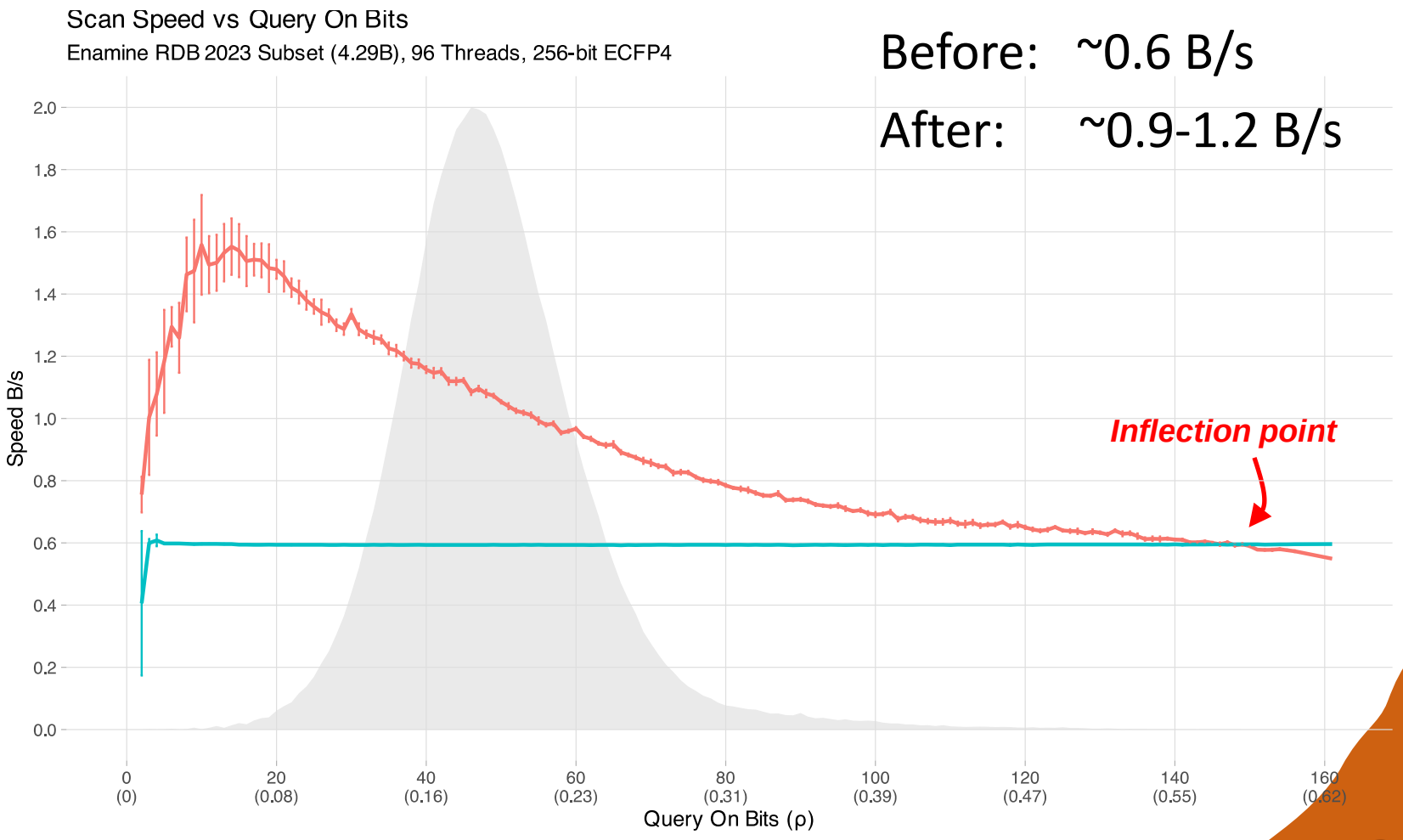


Before: ~0.6 B/s

After: ~0.9-1.2 B/s

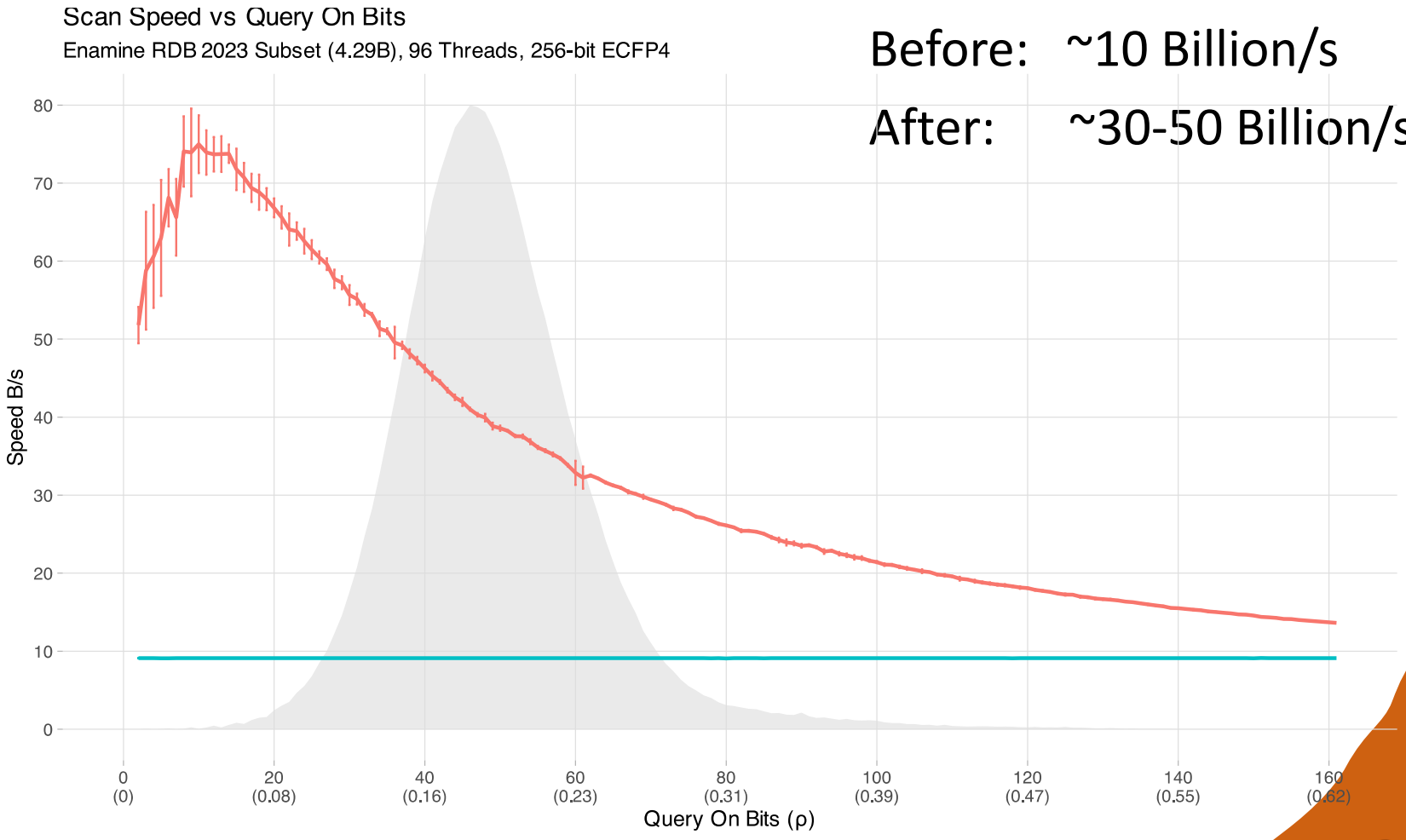
ChEMBL DB POPC distribution shown in grey

SINGLE THREAD REALITY



ChEMBL DB POPC distribution shown in grey

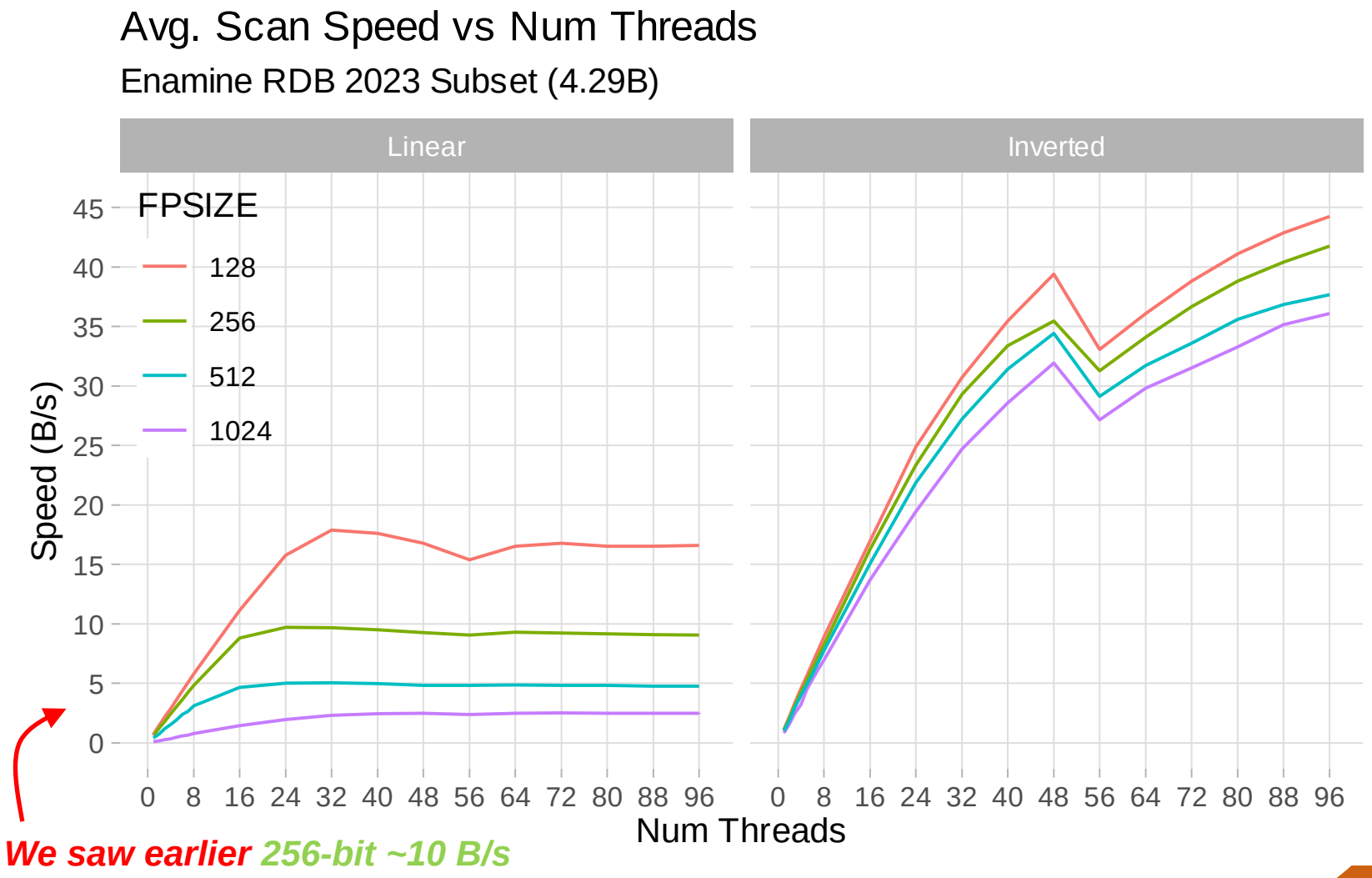
96 THREAD REALITY



Before: ~10 Billion/s

After: ~30-50 Billion/s

THREAD SCALING VS FP SIZE



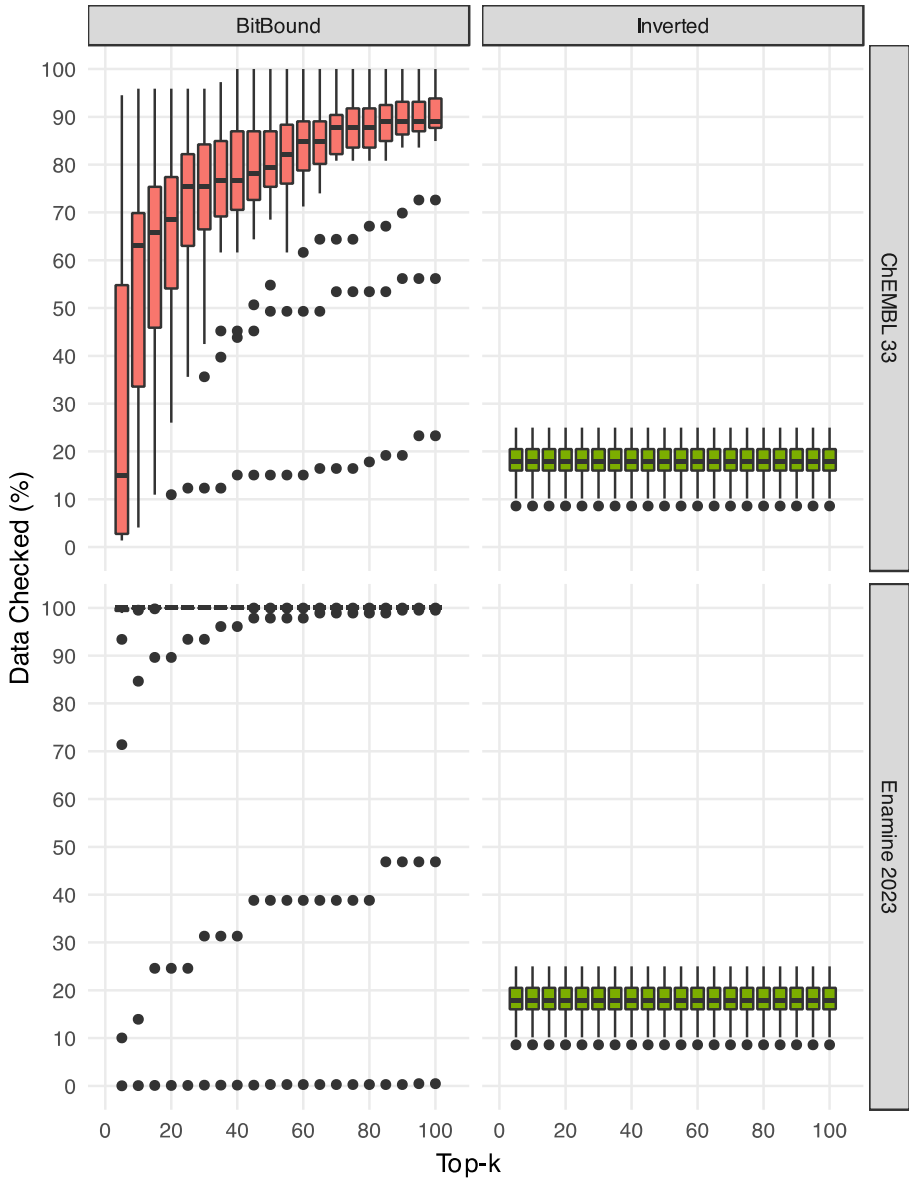
We saw earlier 256-bit ~10 B/s

Queries: 19 top selling small molecules, ECFP4 256-bit
Hardware: 2x AMD EPYC 7443, 1TB DDDR4 @ 3200 MHz

BETTER BOUNDS



COMPARED TO BITBOUND

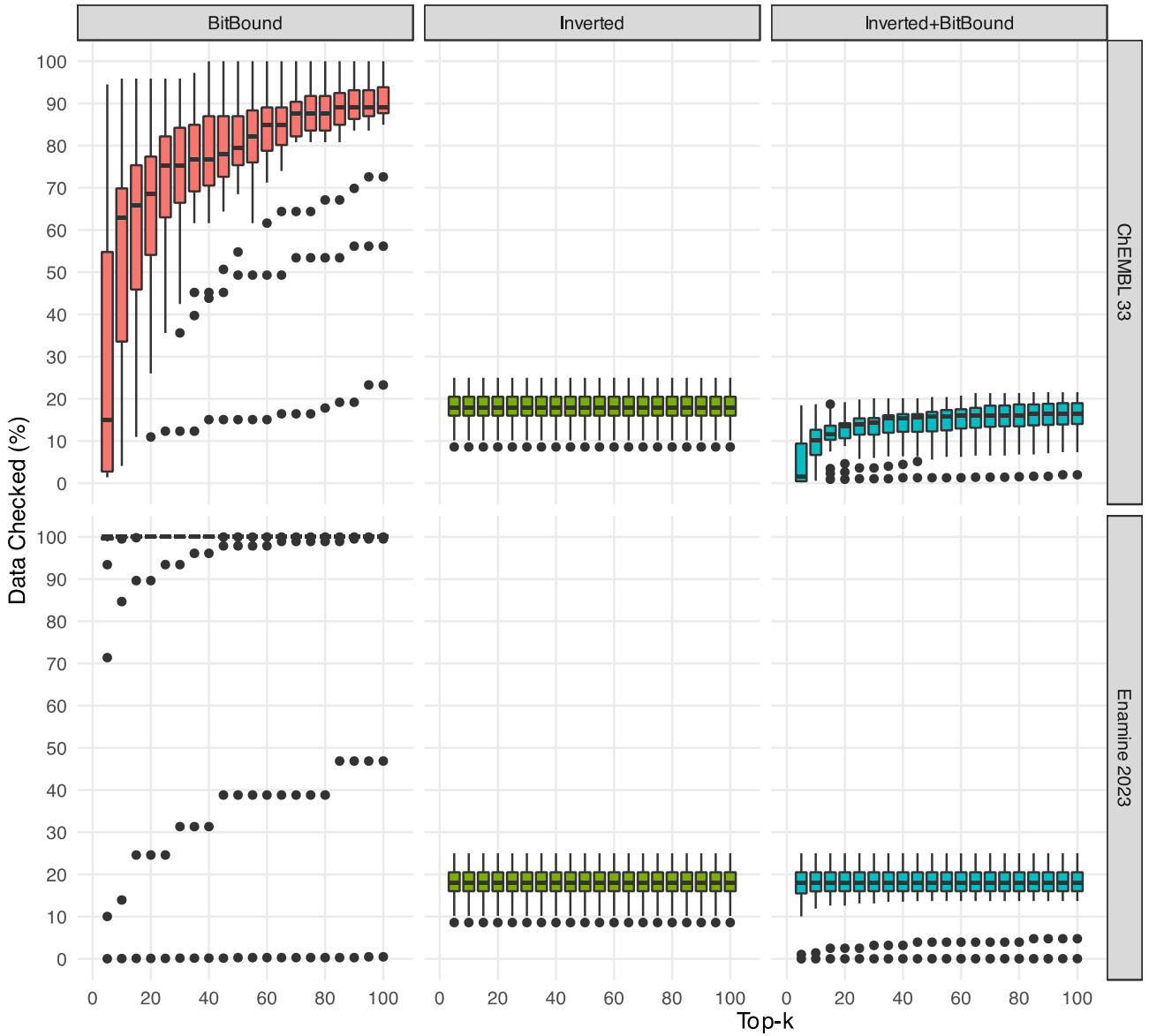


Note **<u>everything is scored</u>** (not just top hits).

BitBound can still be used if you like (next slide) but can do even better



CAN STILL USE BITBOUNDS



PARTIAL SUM BOUNDS

As the inverted bitmaps are added to accumulator we get a better estimation at what the maximum score in a given bucket is.

Consider a query with **50 bits** checking a bucket of records with **52 bits**

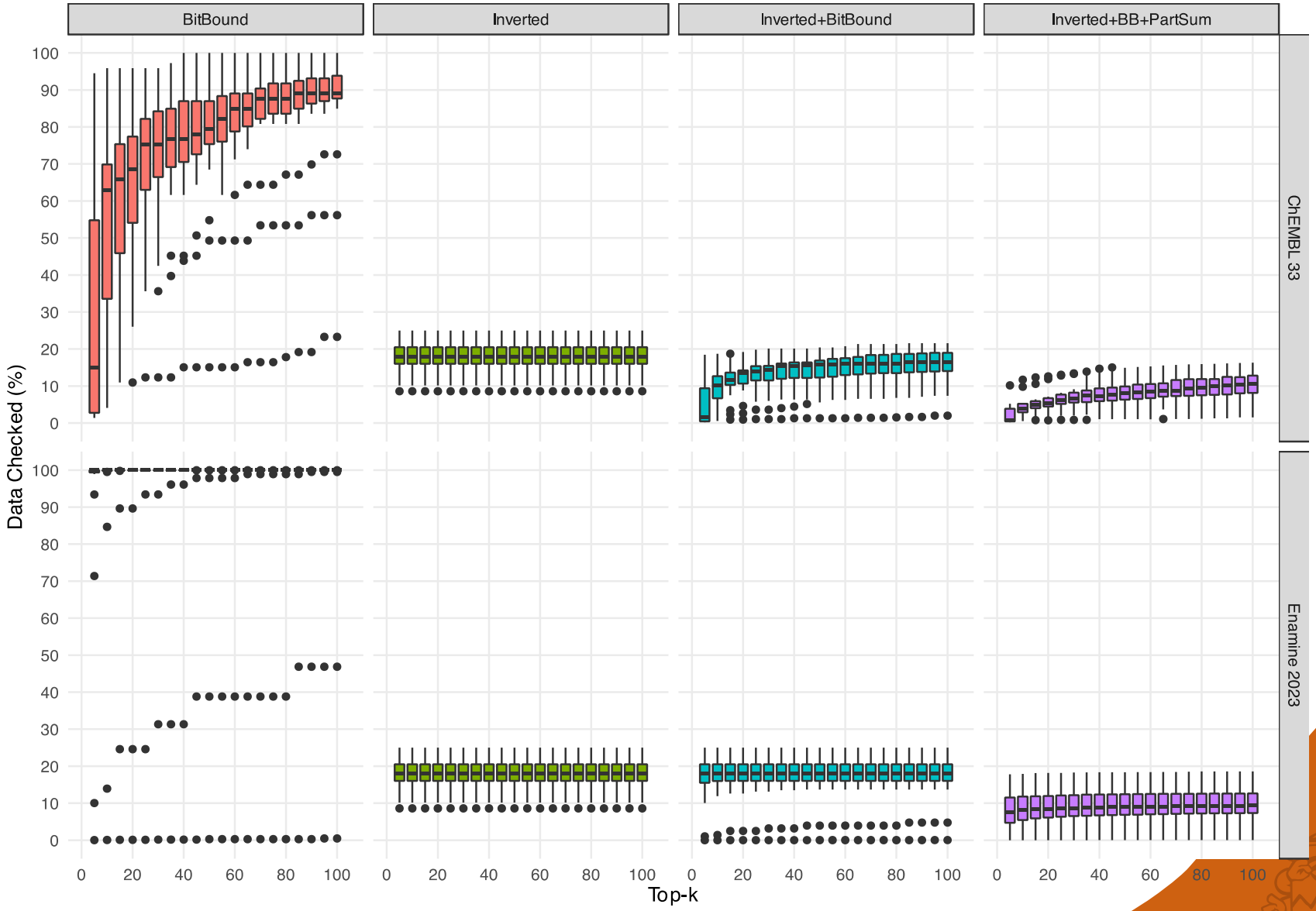
- If we sum 15/50 bitmaps, and find max value seen. If max=4/15, and we have 35/50 left to add. **max_common=39**
  - **PartialSum** method, **t_max ≤ 0.619** (39/(50+52-39))
  - **BitBound** method, **t_max ≤ 0.961** (50/(50+52-50))

Add in rarest bit's first. Similar but more powerful than RISC (2019).

RISC: Rapid Inverted-Index Based Search of Chemical Fingerprints
Jithin Vachery and Sayan Ranu. *Journal of Chemical Information and Modeling* **2019** *59* (6), 2702-2713
DOI: 10.1021/acs.jcim.9b00069



PARTIAL SUM BOUNDS



CONCLUSIONS

- Transposing the bits of a fingerprint it is possible to perform similarity searches that inspect only a small fraction of the database
- Addition of bits requires more work, the reduced memory requirements provide a better balance on modern computer hardware
- More competitive on stricter bottlenecks:
  - disk/network I/O
- Better pruning as the bitmaps are added



FUTURE WORK

- Precomputed addition trees, there is co-occurance between bits. In our addition we could have parts pre-computed/cached
- A query which is similar to a previous set could be handled by a delta and **subtracting** inverted bitmaps
- GPU implementation trivial but **$/GB** less economical
  - DDR4 1TB (16x 64GB ECC@3200MHz) = 2,239 £GBP
  - A100 80GB = 16,670 £GBP
  - H100 80GB = 32,399 £GBP



ACKNOWLEDGEMENTS

Yurii Moroz, Enamine and ChemSpace

John Irwin, ZINC group, UCSF

Andrew Dalke, Dalke Scientific Software

Noel O'Boyle, Heptares

Oliver Korb, Roche

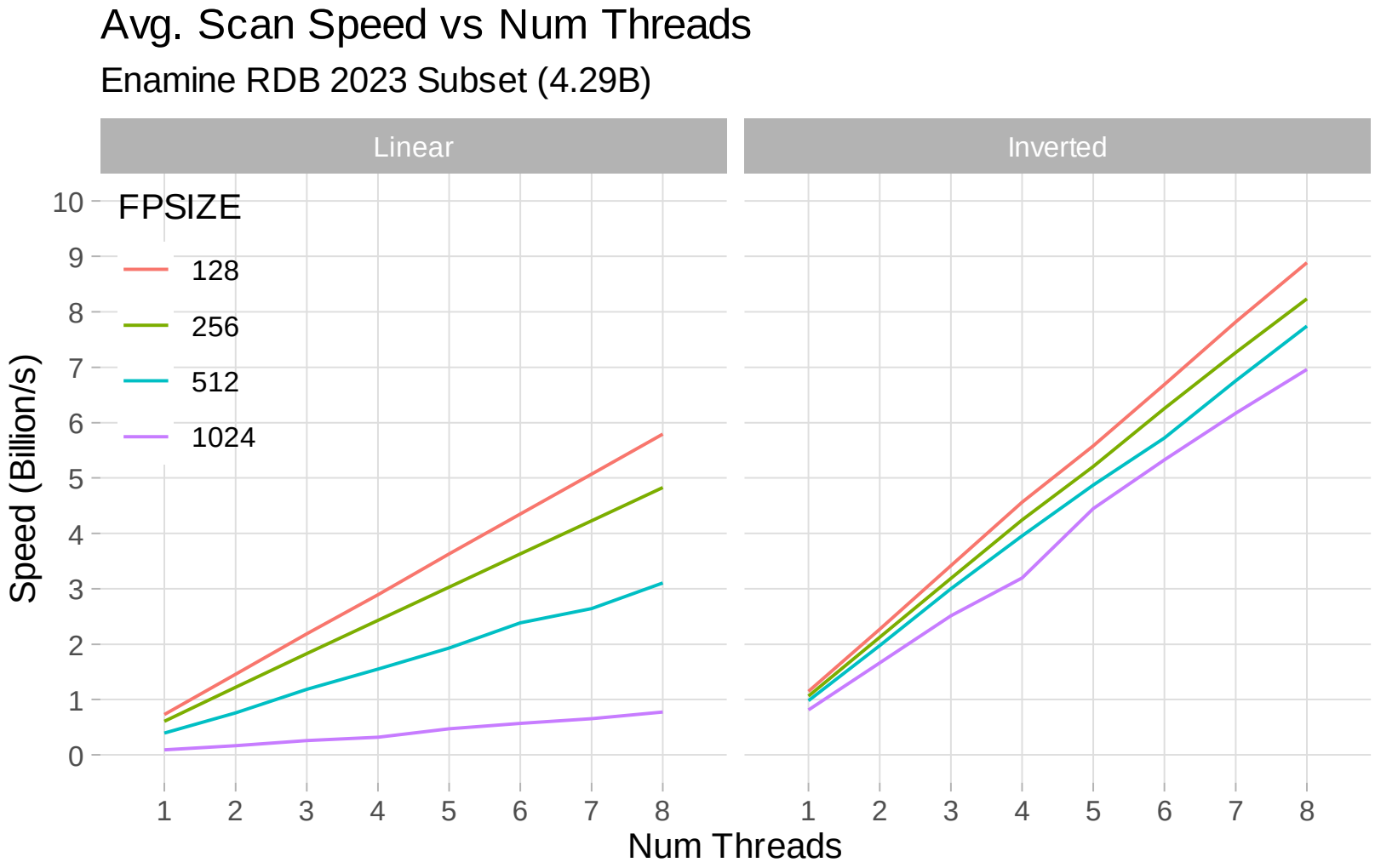


NextMove Software

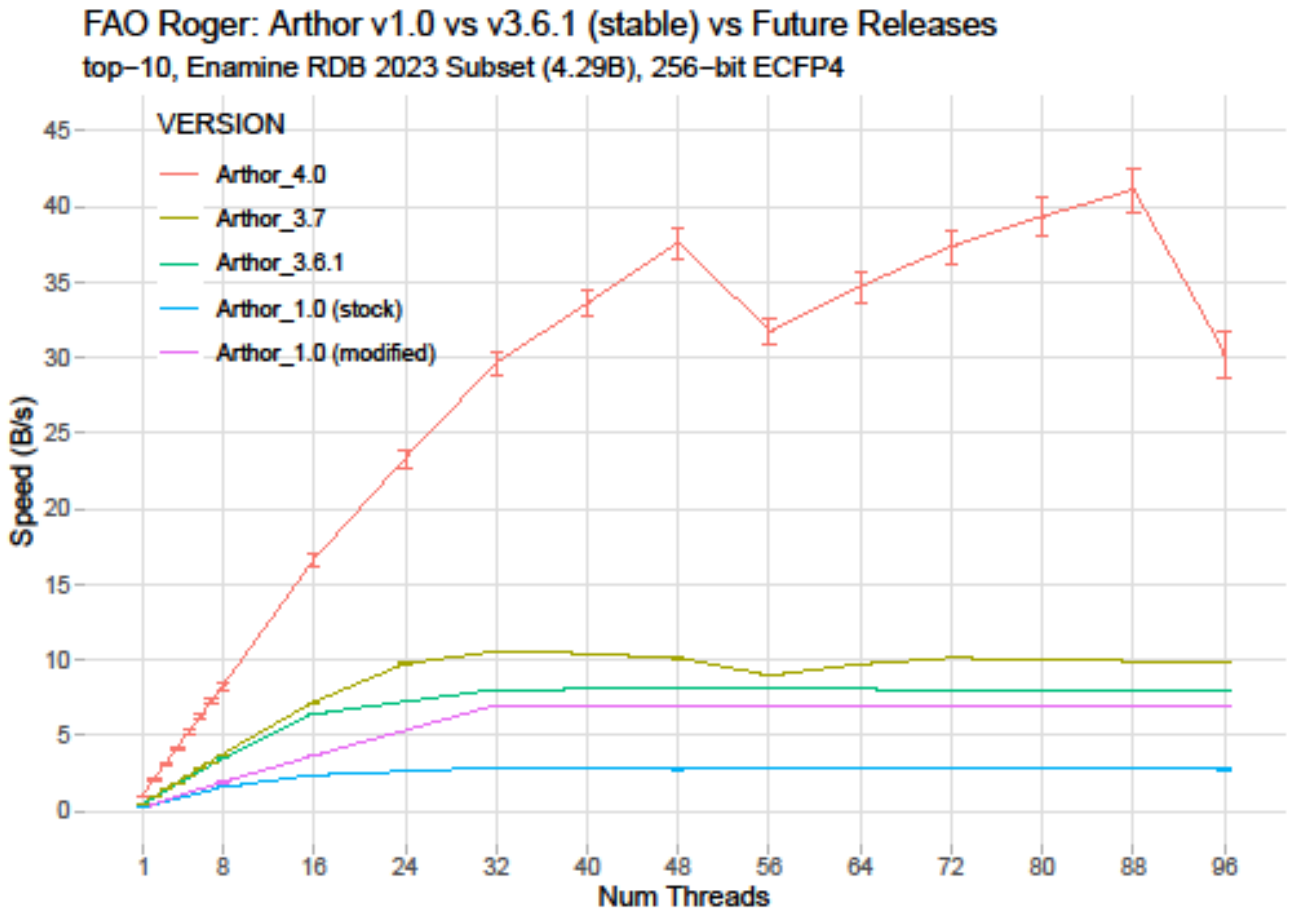
EXTRAS



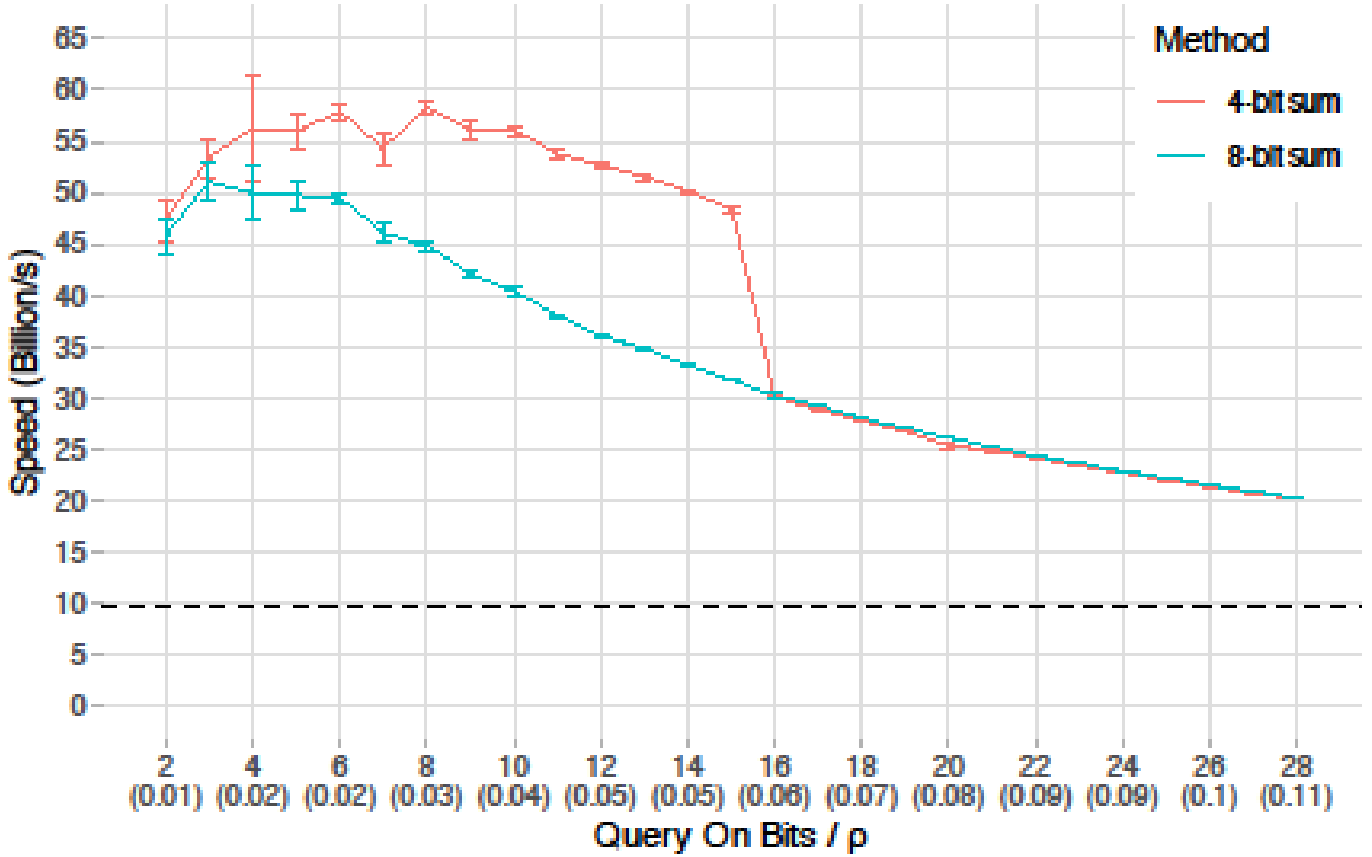
SCALING 1-8


Avg. Scan Speed vs Num Threads
Enamine RDB 2023 Subset (4.29B)
Linear
Inverted
FPSIZE
128
256
512
1024
Speed (Billion/s)
0
1
2
3
4
5
6
7
8
9
10
1
2
3
4
5
6
7
8
Num Threads

ARTHOR VERSION HISTORY

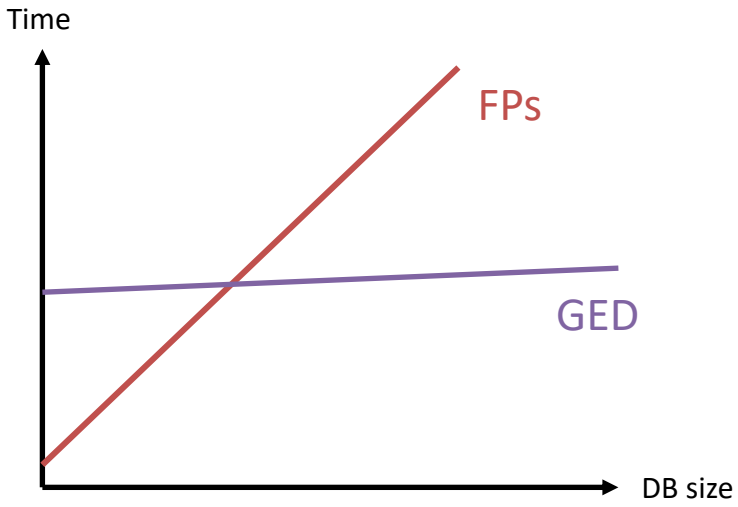


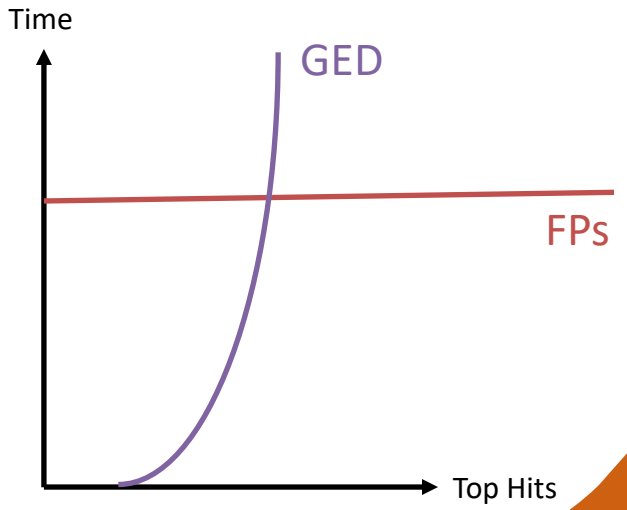
ADDITION TREE 4 VS 8 BIT



FINGERPRINTS SCALE LINEARLY

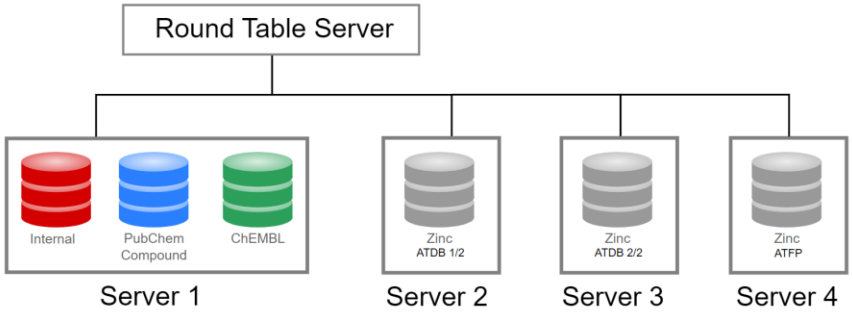
- FPs scale almost linearly ~O(N), where GED is almost constant ~O(1).
- FP results take same time., where GED close neighbors are instant, but distant neighbors take much longer.





FEDERATED SEARCH/MULTIPLE SERVERS

- **RoundTable** is a API that "federates" multiple **Arthor** chemical database (similarity/substructure) servers.
- Because RoundTable's API is the same as Arthor's, RoundTable can recursively federate instances.
  - Care needs to be taken to avoid cycles!



Example Heterogenous Configuration:
ZINC CartBlanche 2023 (35B mols)
Server 1: 1TB RAM (17B mols, 5DBs)
Server 2: 512GB RAM (10B mols, 3DBs)
Server 3: 768GB RAM (4B mols, 1DB)
Server 4: 1TB RAM (4B mols, 1DB)

https://www.youtube.com/watch?v=brxp1x2NV6A



HARDWARE COMPARE

