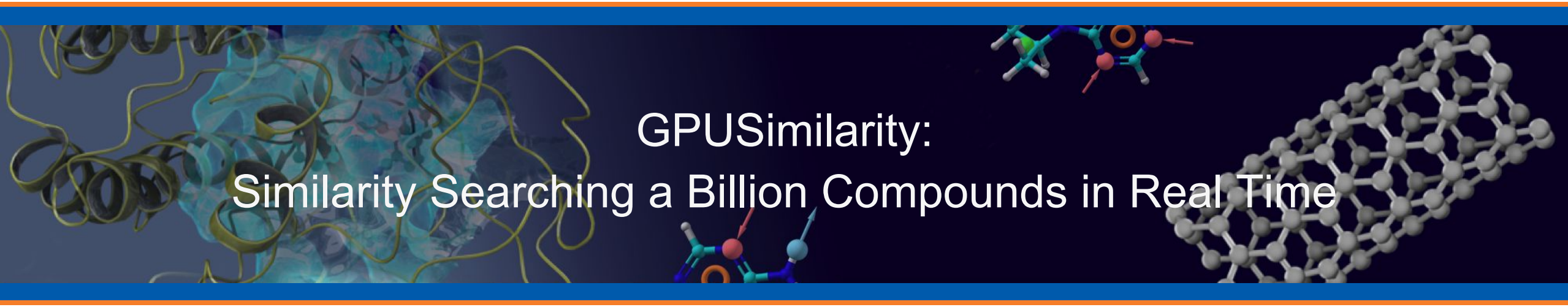


GPUSimilarity:
Similarity Searching a Billion Compounds in Real Time

Pat Lorton

https://github.com/schrodinger/gpusimilarity

Original Idea, and some questions

Original idea:

Computational complexity of fingerprint similarity (tanimoto) is trivial, embarrassingly parallel, and involves no branching. Limiting factor of a straightforward brute force solution is compute power: This sounds like something GPUs would be good at.

Then I asked some questions..

Idea for architecture

This problem has been solved in different ways with various success using complex architectures involving intelligent chemical understanding. This was required because a brute force attempt was simply too expensive in terms of RAM and compute requirements

Re-examining this in 2018, RAM and compute costs have continued to plummet, an architecture around brute-force may now be possible (and preferable).

Reminder: 1 Tesla V100 has a theoretical speed of ***14 terahertz***

128 gigs of very fast ram is purchasable for $1000

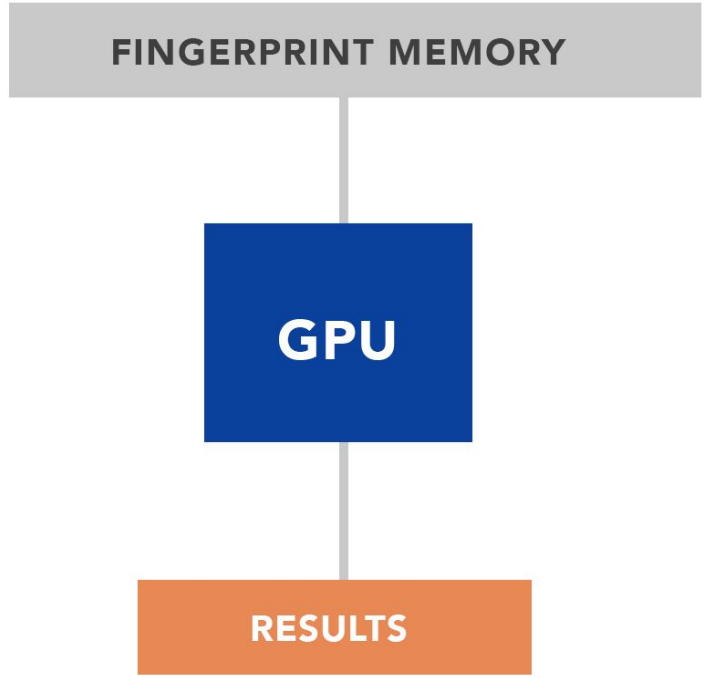
Original super simple architecture

Creating database:

- Use RDKit to generate fingerprints
- Push all fingerprints into one contiguous block of memory
- Use ipyparallel to be able to do this ‘at-scale’

Searching database:

- Read all memory (fingerprint, smiles, ID) into system memory
- Copy only fingerprint memory up to GPU when program launches
- Keep process up listening for requests, execute search on full fingerprint data on GPU
- Sort results on GPU, use index to re-associate with smiles/ID
- Only fetch top results requested (memory-copy-limited)



Original Architecture Retrospective

Successes:

- Very clean code! A couple hundred lines for most work
- Worked great on “small” systems! ~50 million compounds fit on a high end consumer graphics card.
- Search speeds were respectable, Zinc12 Clean library (~17 million compounds) could be searched in 0.05 seconds on a consumer card - an order of magnitude within ‘real time’

Problems...

Let the Complexity Begin..

We ran into several problems as we tried to scale beyond the ‘toy’ case of 20 million compounds:

- Can’t fit everything inside a GPU’s memory (or a couple GPU’s memory)
- Even if you have sufficient RAM, malloc’ing a multi-gigabyte contiguous block of RAM would often fail.
- QByteArray (underlying format used to serialize) could only hold ~2GB each
- Contiguous arrays are relatively hard to parallelize across multiple GPUs

Let the Complexity Begin..

We can into several problems as we tried to scale beyond the ‘toy’ case of 20 million compounds:

- Can’t fit everything inside a GPU’s memory (or a couple GPU’s memory)
- ~~Even if you have sufficient RAM, getting a multi-gigabyte contiguous block of RAM would often fail.~~
- ~~QByteArray (underlying format used to serialize) could only hold ~2GB each~~
- ~~Contiguous arrays are relatively hard to parallelize across multiple GPUs~~

Most of this is solved by slicing memory up into chunks when creating the Database, leaving only one major problem.

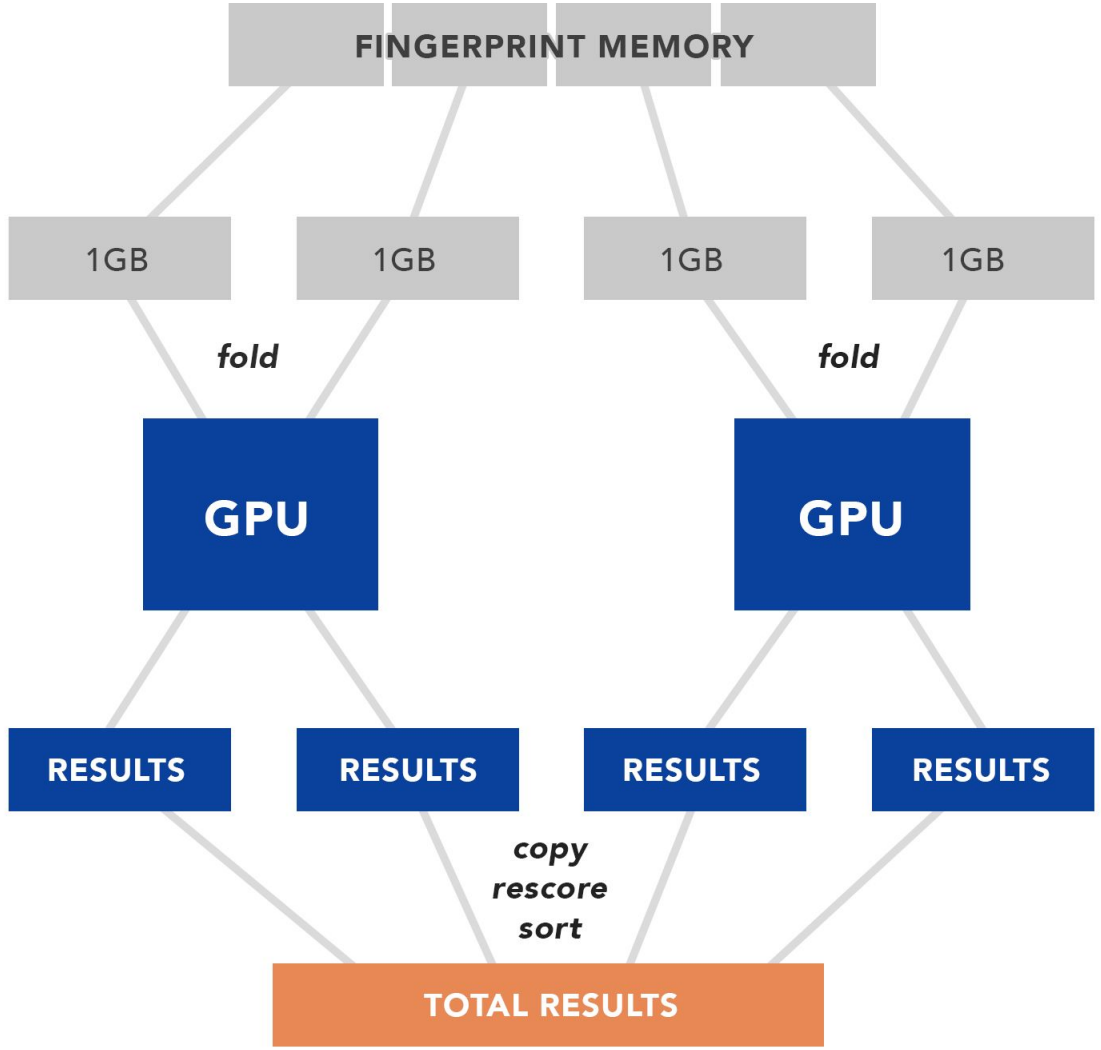
Folding onto the GPU

New architecture idea:

Can we fold the Fingerprints down to always fit current GPU memory, and then re-screen results on CPU using full fingerprints?

Greg ran initial tests using cartridge to confirm (expected) 100% recovery rate using this technique, requiring re-screening ~3x final results for 1024->128bit fold (RDKit Morgan Fingerprints).

This works for bringing back results within an expected cutoff, but what if you only want the top N results? Bringing back too much memory from GPUs is rate limiting.



SCHRÖDINGER.

Worries of folding

The top 20 folded results are likely not the exact top 20 unfolded results, how to deal with this?

Bring back a larger number of results and re-score on the CPU

How big is ‘a larger number’?

Worries of folding

The top 20 folded results are likely not the exact top 20 unfolded results, how to deal with this?

Bring back a larger number of results and re-score on the CPU

How big is ‘a larger number’?

I made something up! Trial and error lead to:

$(\text{Number of requested results}) * N\log_2 2N$ -- N=Folding Factor

How did this do?

Results of bringing back fixed results

Folding Level	Mismatch %	Mismatches >0	Mismatches >0.5	Mismatches >0.7
2	0	0	0	0
4	0.1%	2	0	0
8	1.45%	29	1	0

*Results collected with RDKit Morgan Fingerprints against Zinc12 clean

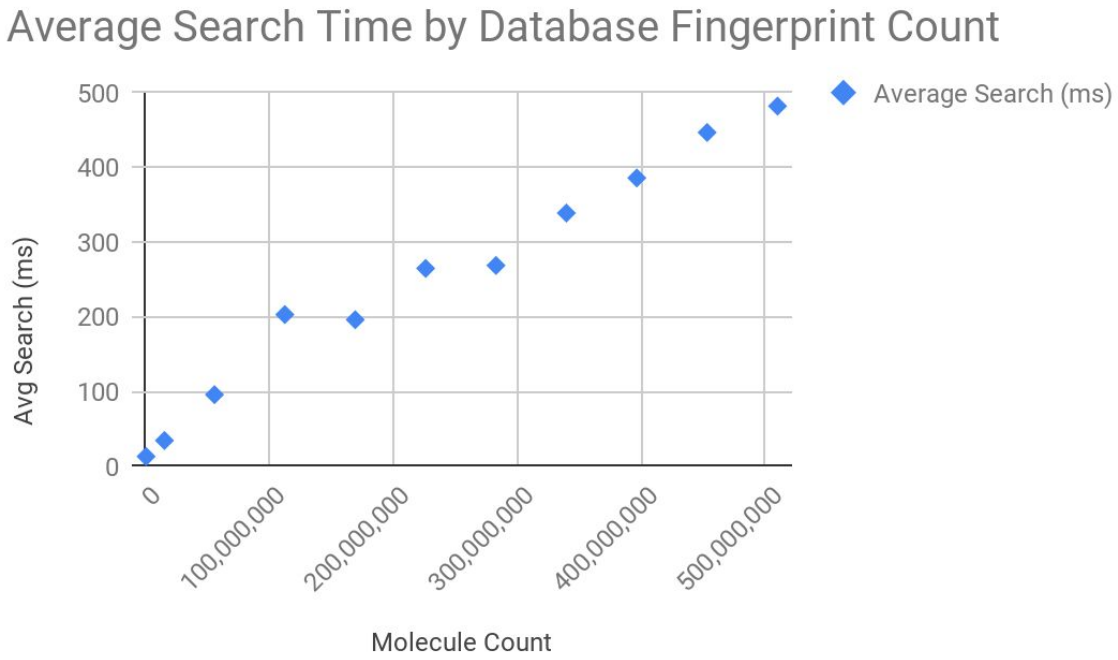
Top 20 results of 100 different searches against 17M compounds
Fingerprints are 1024 bits, folding level indicates new fingerprint sizes of 512, 256 and 128 respectively.

Results were worse with just ‘N’, NlogN prevents many mismatches

So now we can scale, how did we do?

"Hobby Machine": 6-core i7, 128gb ram, 2 GeForce 1080Ti. <$4000 to build

Data set	Molecule Count	100 searches (ms)	Fold Factor	Average Search (ms)	Scaled to billion (ms)
Chembl	1,618,358	1279	1	12.79	7903.07
Zinc12	16,403,849	3417	1	34.17	2083.05
Enamine 1/12	56,667,620	9549	1	95.49	1685.09
Enamine 2/12	113,335,291	20276	1	202.76	1789.03
Enamine 3/12	170,002,932	19585	2	195.85	1152.04
Enamine 4/12	226,670,577	26458	2	264.58	1167.24
Enamine 5/12	283,338,230	26851	4	268.51	947.67
Enamine 6/12	340,005,838	33860	4	338.6	995.87
Enamine 7/12	396,673,470	38560	4	385.6	972.08
Enamine 8/12	453,341,072	44654	4	446.54	985.00
Enamine 9/12	510,008,695	48177	8	481.77	944.63

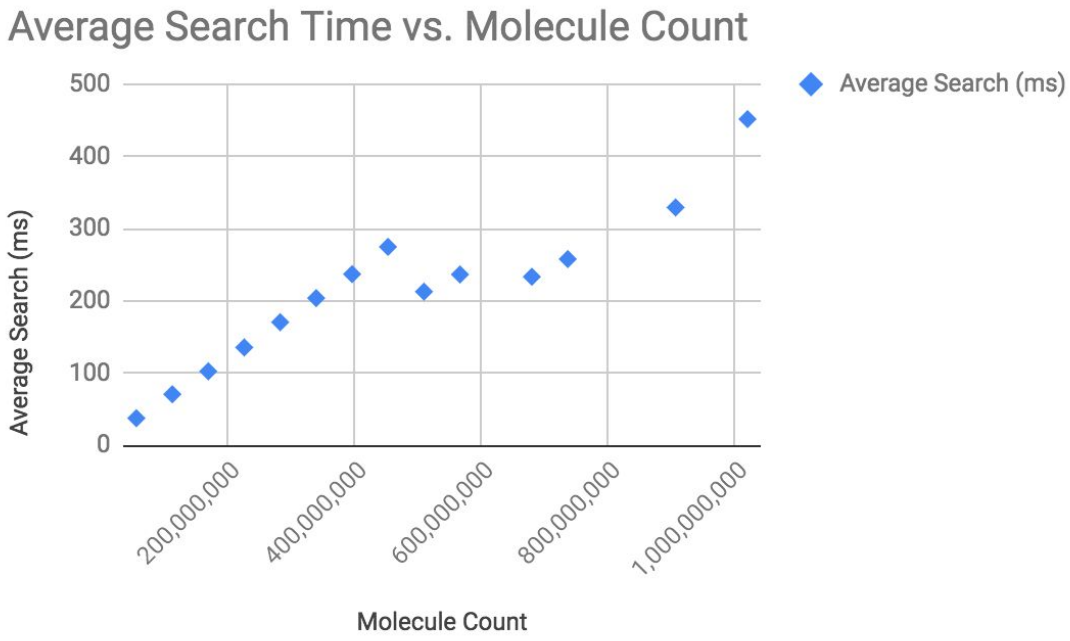


SCHRÖDINGER.

So now we can scale, how did we do?

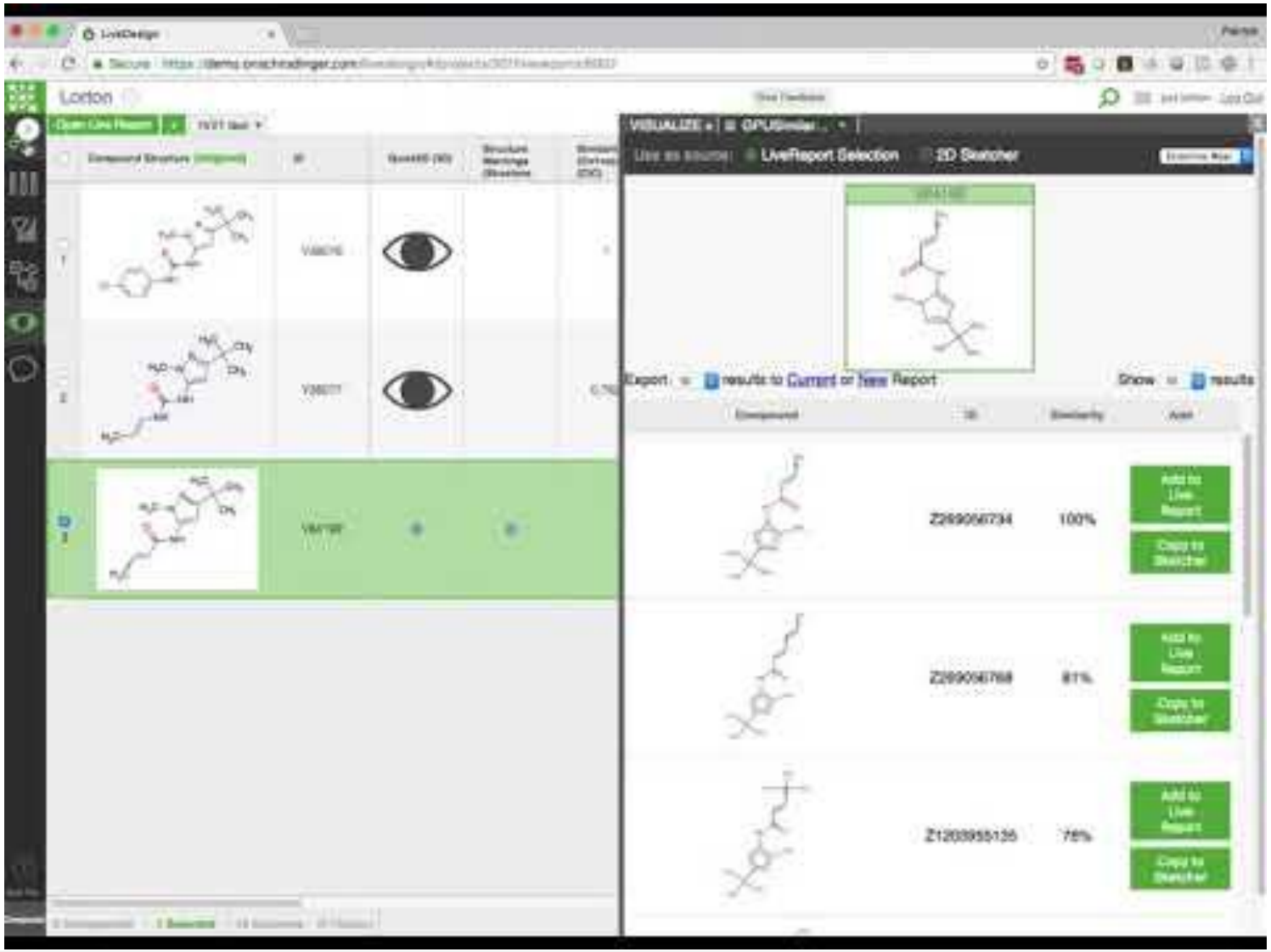
Server-class: 32-core i7, 256gb ram, 4 Tesla V100. A lot to build, even more on Amazon...

Data set	Molecule Count	100 searches (ms)	Fold Factor	Average Search (ms)	Scaled to billion (ms)
Enamine 1/12	56,667,620	3757	1	37.57	662.99
Enamine 2/12	113,335,291	7059	1	70.59	622.84
Enamine 3/12	170,002,932	10241	1	102.41	602.40
Enamine 4/12	226,670,577	13559	1	135.59	598.18
Enamine 5/12	283,338,230	17042	1	170.42	601.47
Enamine 6/12	340,005,838	20375	1	203.75	599.25
Enamine 7/12	396,673,470	23708	1	237.08	597.67
Enamine 8/12	453,341,072	27477	1	274.77	606.10
Enamine 9/12	510,008,695	21269	2	212.69	417.03
Enamine 10/12	566,676,374	23656	2	236.56	417.45
Enamine 12/12	680,011,634	23328	2	233.28	343.05



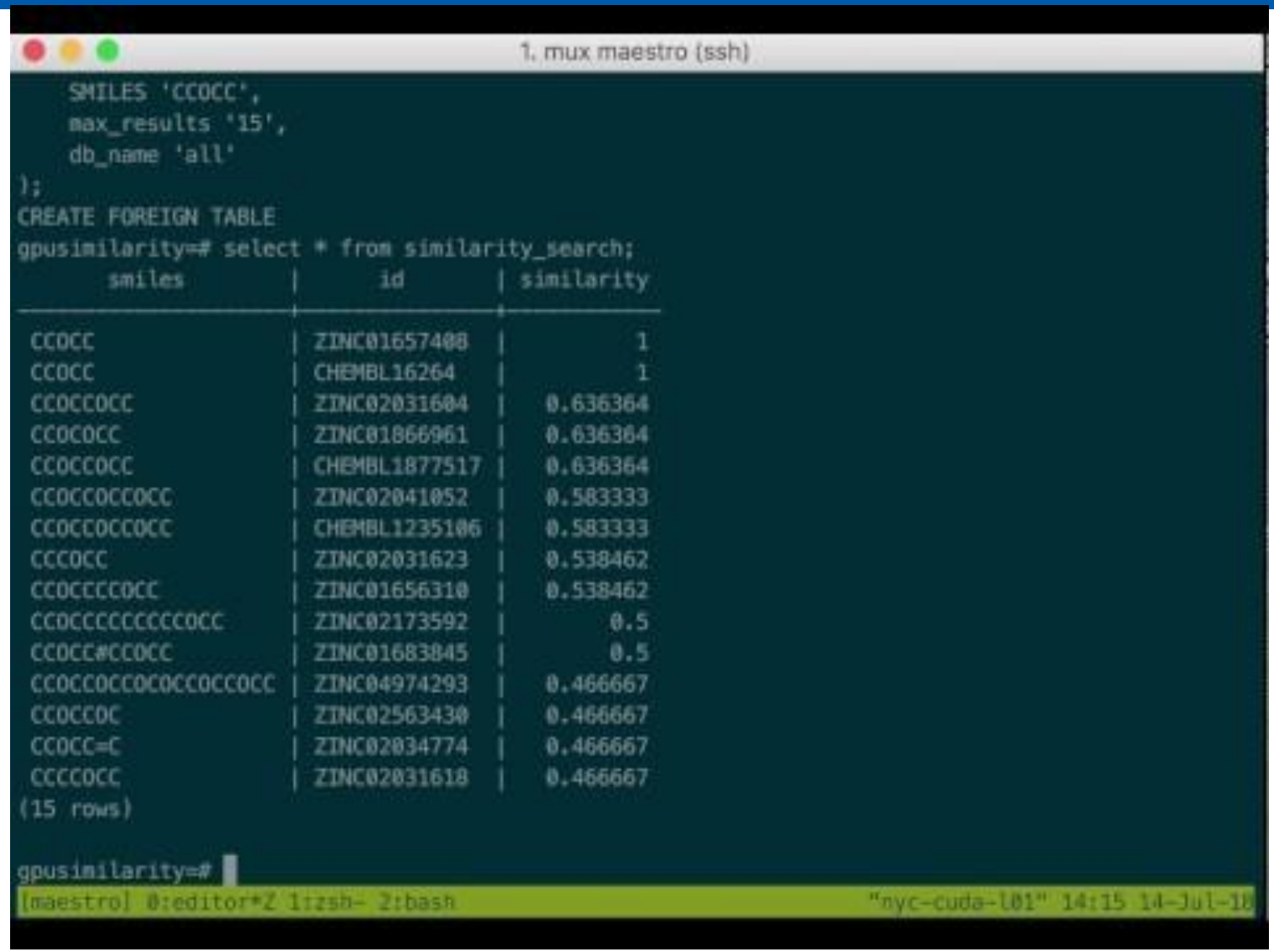
SCHRÖDINGER.

How do those numbers look in practice?



SCHRÖDINGER.

How does this look from the database..



SCHRÖDINGER.

One last fun test...

Amazon has a node with 8 Tesla V100's that can actually fit a billion unfolded compounds in GPU memory. How fast can we brute force it? Hopefully a preview of the future…

One last fun test...

Amazon has a node with 8 Tesla V100's that can actually fit a billion unfolded compounds in GPU memory. How fast can we brute force it? Hopefully a preview of the future…

Searching 1 billion compounds takes 0.35 seconds. Hmm.

Conclusions

- Compute speeds and RAM prices have come down enough that brute-force is a reasonable method for a similarity search server. It will only become cheaper.
- Fingerprint folding with re-scoring is a reasonable method to deal with lower GPU memory than CPU memory.
- It’s a race to see whether GPU’s can add more memory faster than easily-purchasable compound libraries will grow.

Further research and general to-do

Minimum fold during brute force?

Better math for choosing how many results to bring back off GPU

Try a lot more fingerprints for validation

Think about merging and sorting results of chunk searches on GPU

Port fingerprint folding to GPU

An int-less future..