Lindsey Stuff

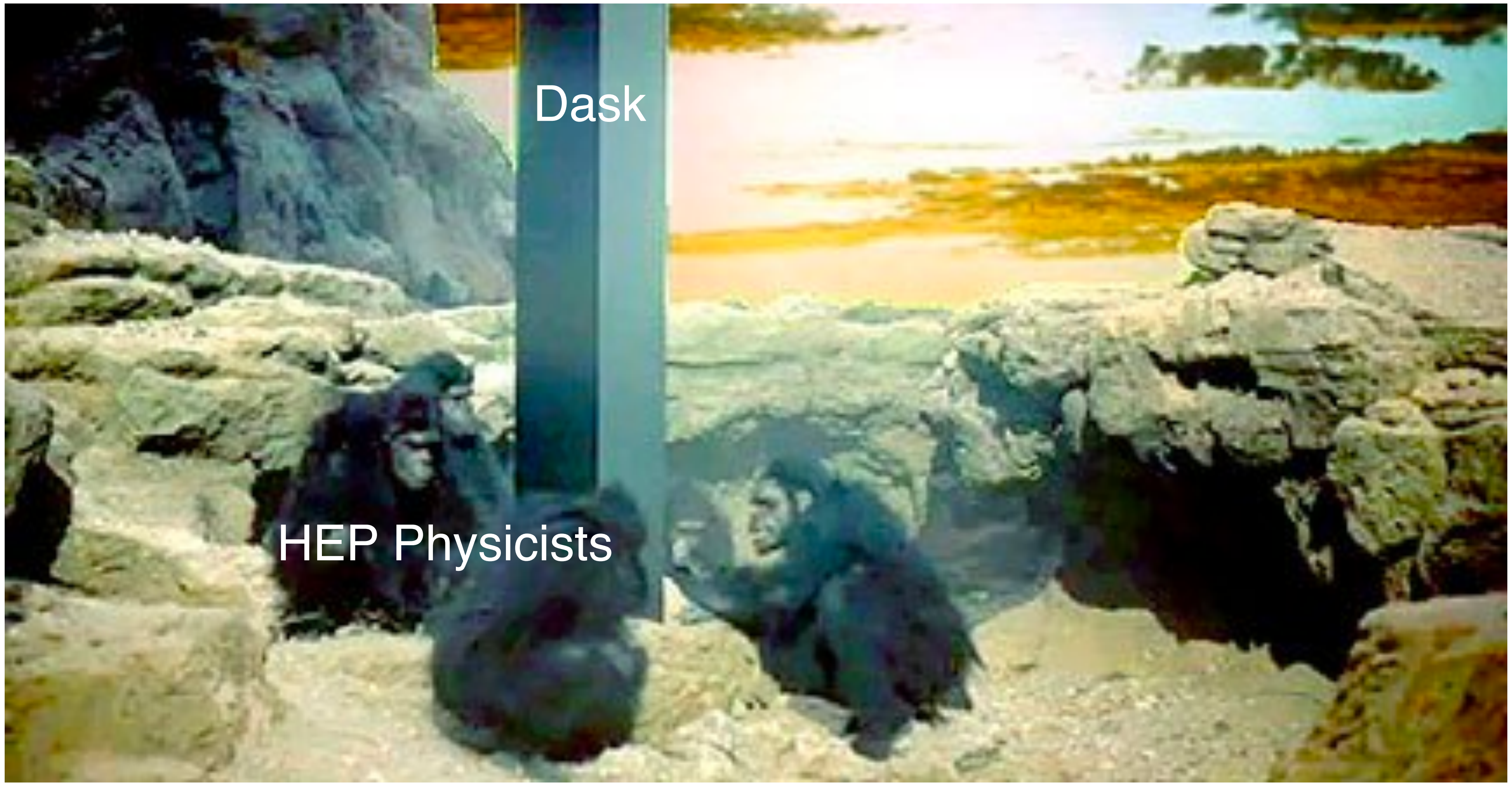
Lindsey Gray
PyHEP.dev 2023
25 July 2023

About Me!

- Fermilab - Staff Scientist
- Physics
  - Multi-Vector Boson physics, usually involving a photon, occasionally jet final states
  - EFT measurements past and present, from modified vertex functions to the more refined modern approaches
  - Occasional forays into final states with boosted jets (didn't really stick)
- Software
  - CMS E/gamma Reconstruction and Particle Flow
  - End-to-end HGCAL reconstruction with graph neural networks
  - Coffea (see Nick's intro) - more recently coffea 2023 dask migration
  - Infrastructure in/around analysis facilities:
    - Nvidia triton, dask/distributed, dask-awkward, dask-histogram, caches, network scheduling
- Hardware(-ish)
  - Smartpixels (neural networks in asics for on-pixel-sensor reconstruction)
    - Finding efficient neural networks that can produce understood error predictions
  - Precision timing detectors (CMS MIP Timing Detector)
    - Developed 4D vertexing algorithm, shaped physics case, initial detector design considerations, technical proposal, TDR, beam tests

Dask!

Approximately November 2022:

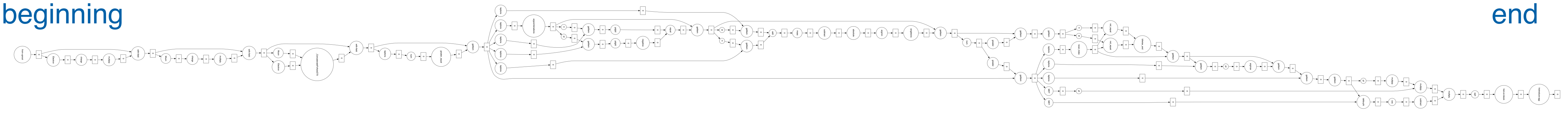


Dask!

A few weeks later:

Dask - more seriously

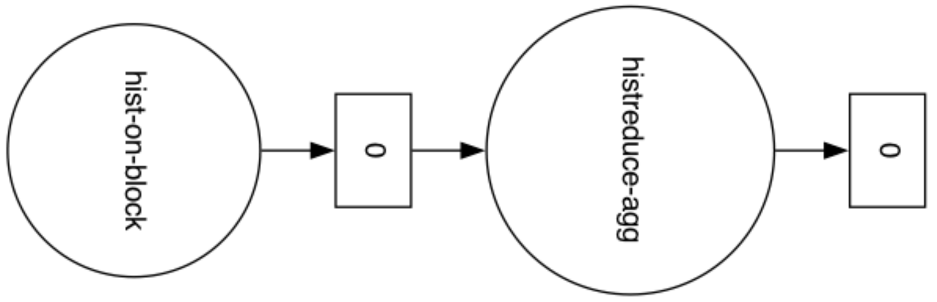
query beginning



query end

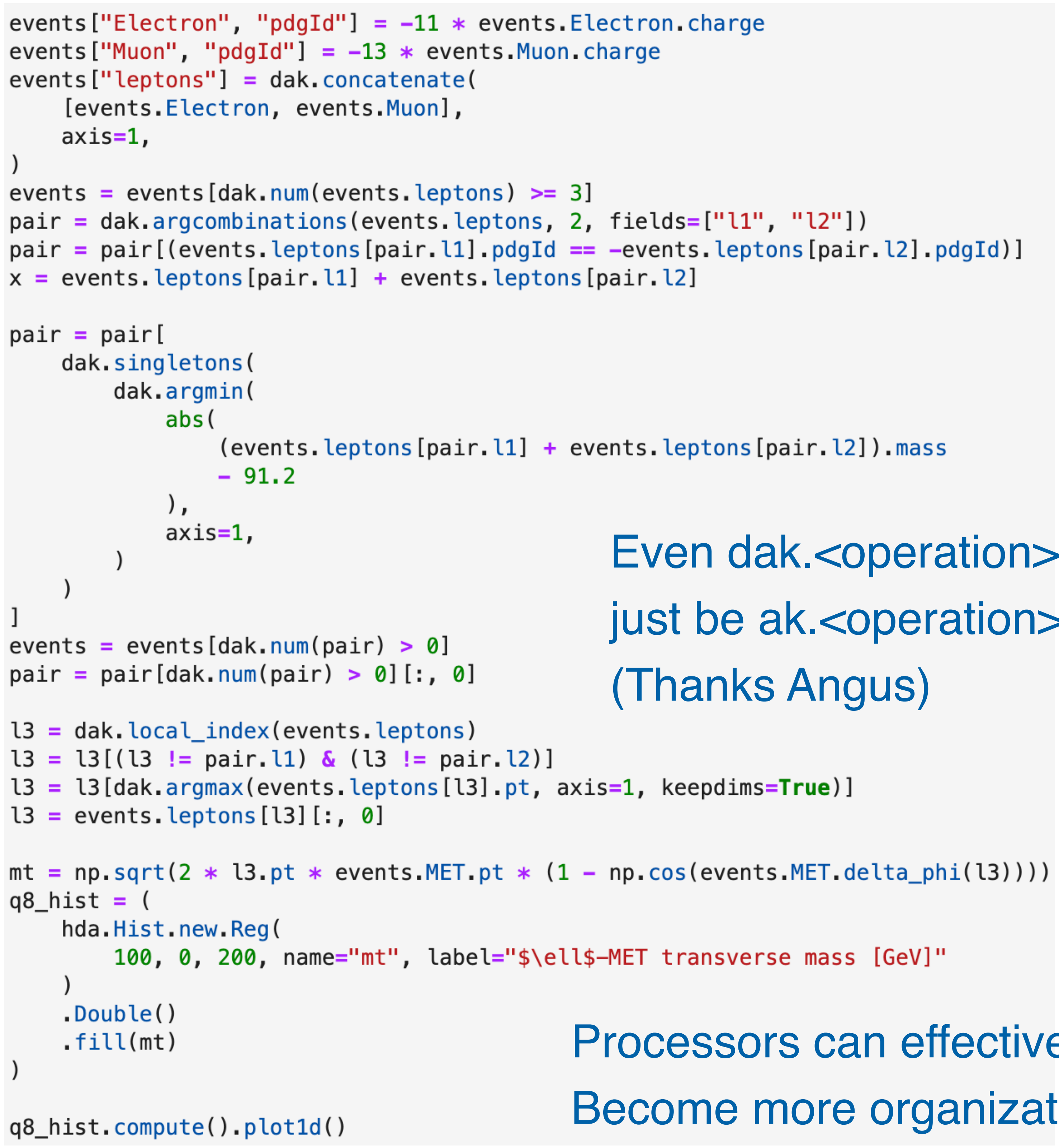
dask.optimize(q8_hist)

query beginning



query end

Dask - the code looks the same!



```
events["Electron", "pdgId"] = -11 * events.Electron.charge
events["Muon", "pdgId"] = -13 * events.Muon.charge
events["leptons"] = dak.concatenate(
    [events.Electron, events.Muon],
    axis=1,
)
events = events[dak.num(events.leptons) >= 3]
pair = dak.argcombinations(events.leptons, 2, fields=["l1", "l2"])
pair = pair[(events.leptons[pair.l1].pdgId == -events.leptons[pair.l2].pdgId)]
x = events.leptons[pair.l1] + events.leptons[pair.l2]

pair = pair[
    dak.singletons(
        dak.argmin(
            abs(
                (events.leptons[pair.l1] + events.leptons[pair.l2]).mass
                - 91.2
            ),
            axis=1,
        )
    )
]
events = events[dak.num(pair) > 0]
pair = pair[dak.num(pair) > 0][:, 0]

l3 = dak.local_index(events.leptons)
l3 = l3[(l3 != pair.l1) & (l3 != pair.l2)]
l3 = l3[dak.argmax(events.leptons[l3].pt, axis=1, keepdims=True)]
l3 = events.leptons[l3][:, 0]

mt = np.sqrt(2 * l3.pt * events.MET.pt * (1 - np.cos(events.MET.delta_phi(l3))))
q8_hist = (
    hda.Hist.new.Reg(
        100, 0, 200, name="mt", label="$\ell$-MET transverse mass [GeV]"
    )
    .Double()
    .fill(mt)
)

q8_hist.compute().plot1d()
```

Even dak.<operation> can now just be ak.<operation>, so transition is easy.
(Thanks Angus)

Processors can effectively disappear?
Become more organizational?

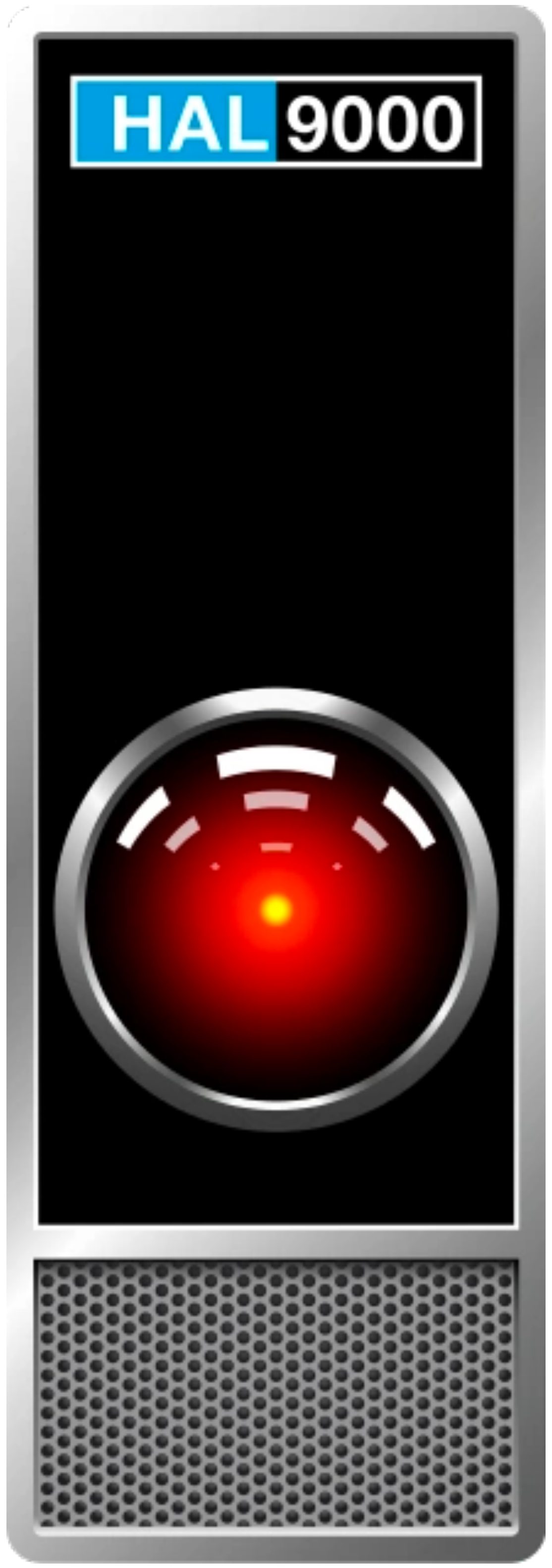
Present status

- Looks like ~end of August we'll have a proper release
  - Given what needs to be accomplished seems reasonable!
- Making sure we match user expectations and ensure a soft landing will be important in this period
  - It's hard to convince people to change an in-flight analysis until it's approved
  - However we need significantly change documentation that is also much better
    - raw awkward -> dask-awkward (or delayed compute in general) is not *really* conceptually trivial!

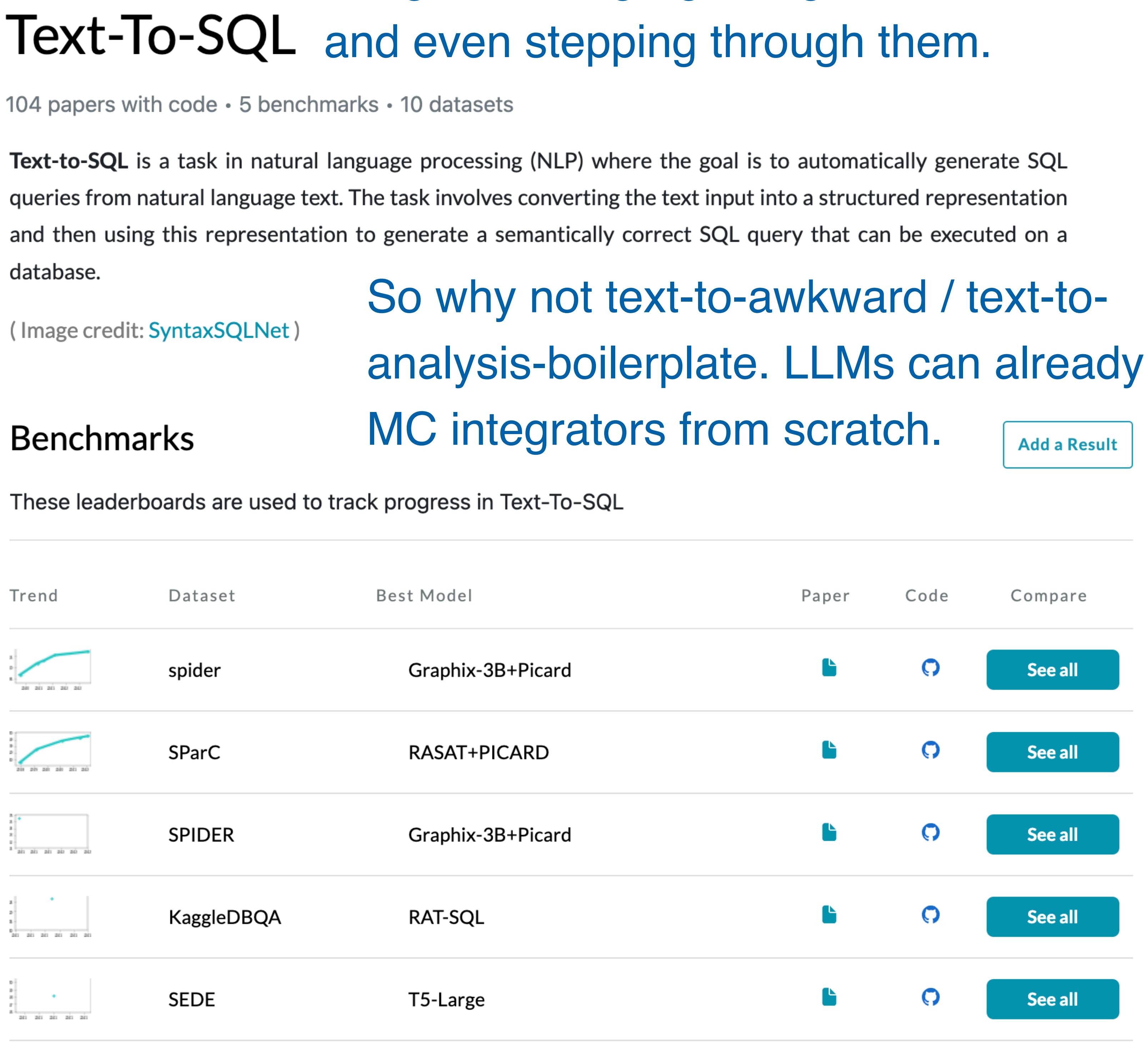
What's Next / This Workshop?

- It's time to start thinking about achieving the most we can at scale
  - We have achieved 2hr turnaround for run 2 analysis (with predicate pushdown as skims)
  - We need just-in-time predicate pushdown (servicex? How do we make using it smooth?)
  - We need histograms distributed across cluster memory (50GB histograms anyone?)
  - We need to make sure we can move from ML training to deployment smoothly (4B event trainings?)
- What are the *interfaces* we expose to users to make them most effective and flexible?

What about llamas?



We can and **should** exploit large language models for documentation and even code. LLMs are great at regurgitating well known concepts and even stepping through them.



Text-To-SQL

104 papers with code • 5 benchmarks • 10 datasets

Text-to-SQL is a task in natural language processing (NLP) where the goal is to automatically generate SQL queries from natural language text. The task involves converting the text input into a structured representation and then using this representation to generate a semantically correct SQL query that can be executed on a database.

(Image credit: SyntaxSQLNet)

Benchmarks

These leaderboards are used to track progress in Text-To-SQL

Trend	Dataset	Best Model	Paper	Code	Compare
	spider	Graphix-3B+Picard			See all
	SParC	RASAT+PICARD			See all
	SPIDER	Graphix-3B+Picard			See all
	KaggleDBQA	RAT-SQL			See all
	SEDE	T5-Large			See all

So why not text-to-awkward / text-to-analysis-boilerplate. LLMs can already write MC integrators from scratch.

Accurate depiction of *the future*