

Integrating LLM-Powered Analysis Assistants into Existing Analytical Workflows: An Odyssey



Connor Scully-Allison

Invited Talk, 09.01.2026

GoAID

About Me

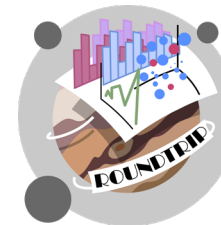
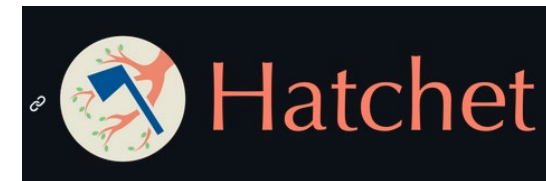
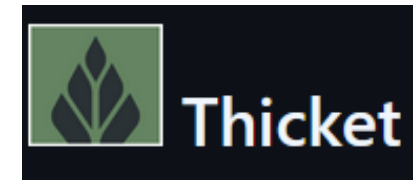


PhD, University of Utah, 2025

- Data Visualization
- Focus on Integrating Visualization and Code Environments



Postdoctoral Scholar,
University of Chicago



About Me



PhD, University of Utah, 2025

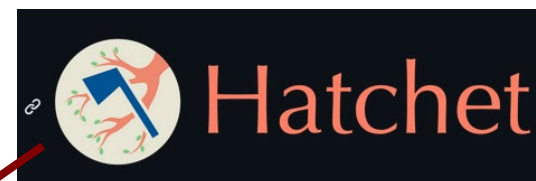
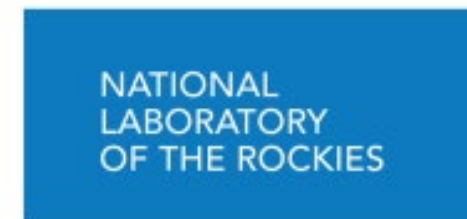
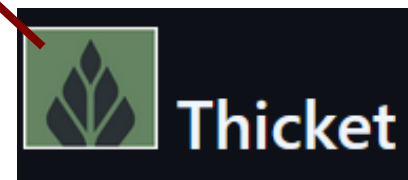
- Data Visualization
- Focus on Integrating Visualization and Code Environments



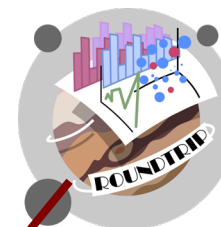
Postdoctoral Scholar,
University of Chicago



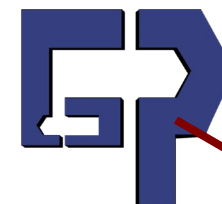
HPDC 23'



HUST 20'
E-Science 22'



TVCG 24'



PEARC 24'
TVCG 26'

Cast – The Domain Scientist



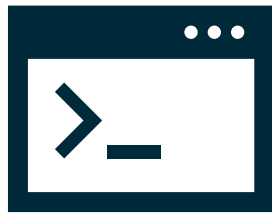
Cast – The HPC Researchers



Cast – The HPC Researchers



The Data They Use



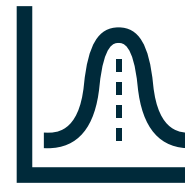
Code



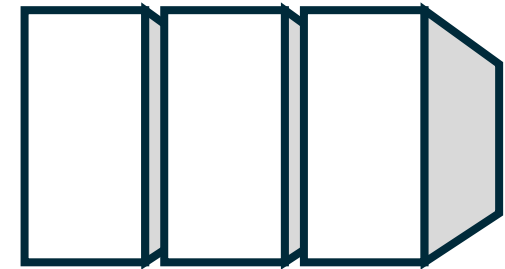
Performance Data



Verification Data

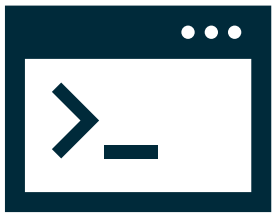


Telemetry Data



The Human

Data Gap



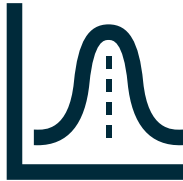
Code



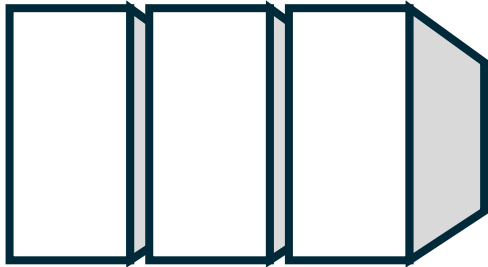
Performance Data



Verification Data

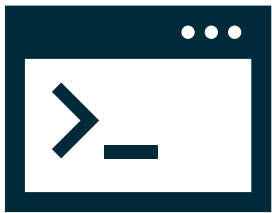


Telemetry Data



The Human

Data Gap



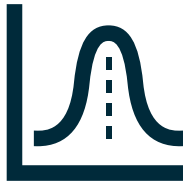
Code



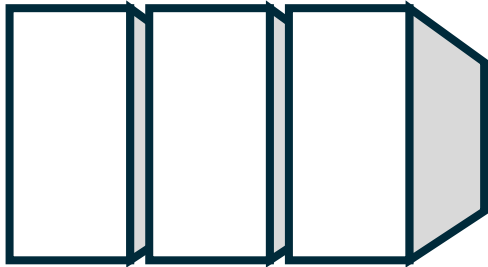
Performance Data



Verification Data



Telemetry Data

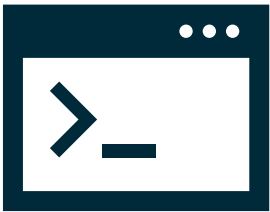
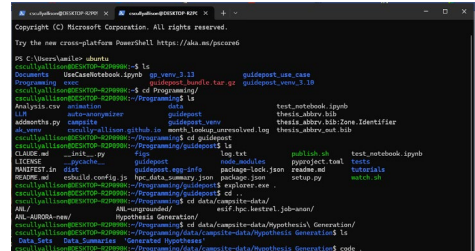
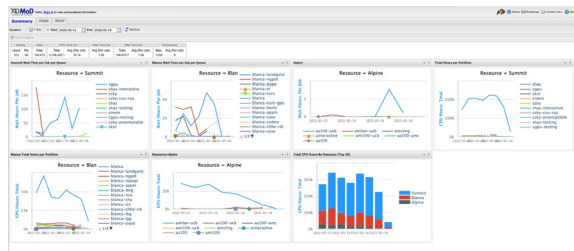
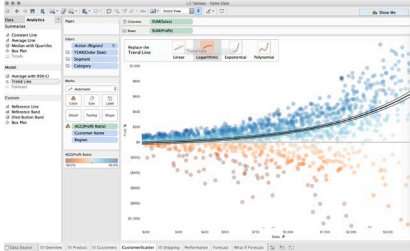
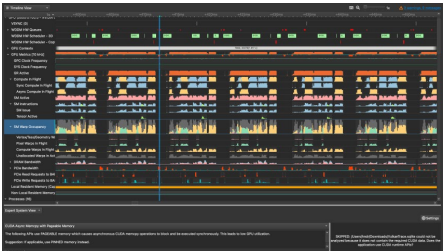
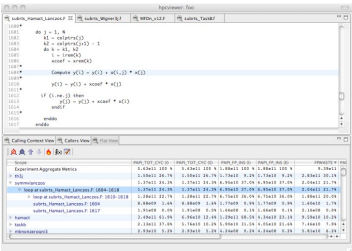


The Human

Data Gap

Performance Engineer

Operational Data Analyst



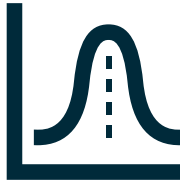
Code



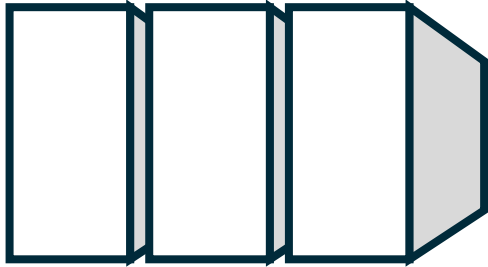
Performance Data



Verification Data



Telemetry Data



The Human

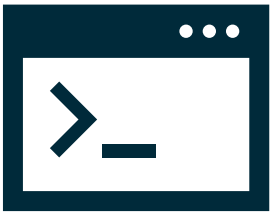
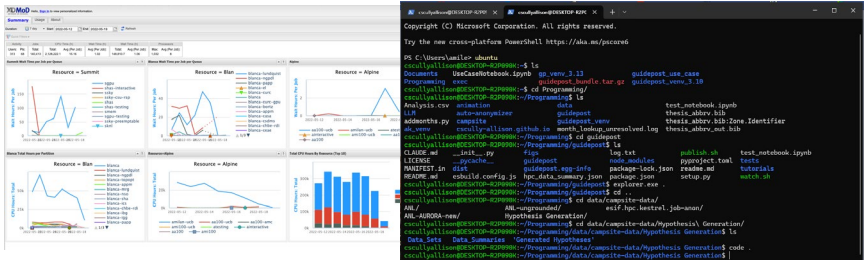
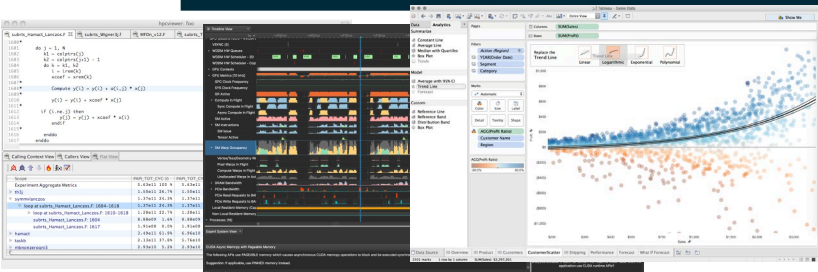
Data Gap



Performance Engineer



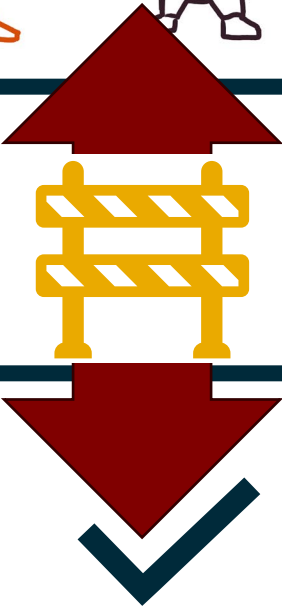
Operational Data Analyst



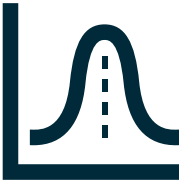
Code



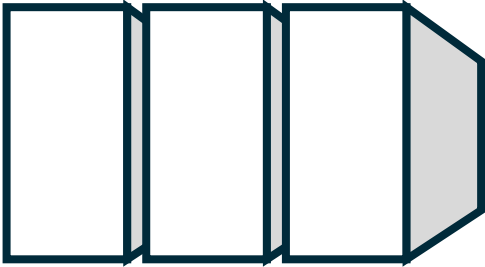
Performance Data



Verification Data



Telemetry Data



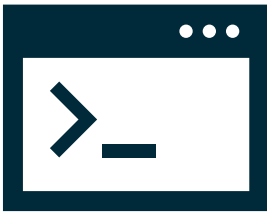
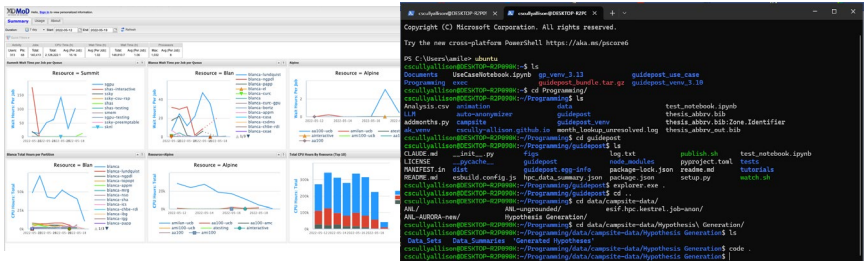
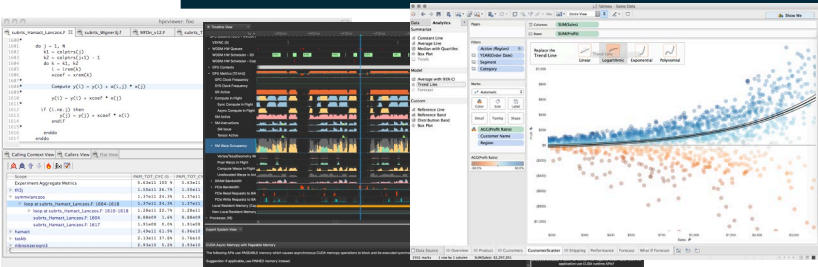
The Human

Data Gap

Performance Engineer

Operational Data Analyst

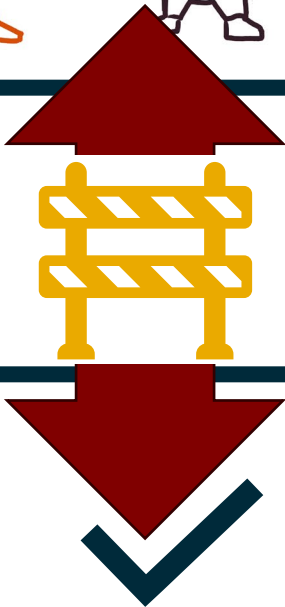
Tooling Gap



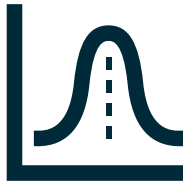
Code



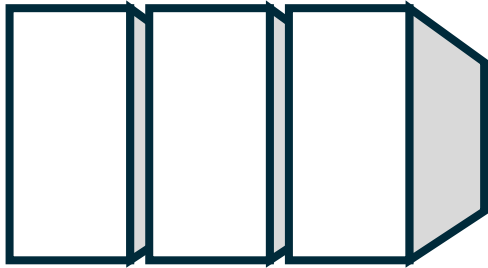
Performance Data



Verification Data



Telemetry Data



The Human

Data Gap



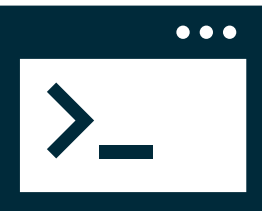
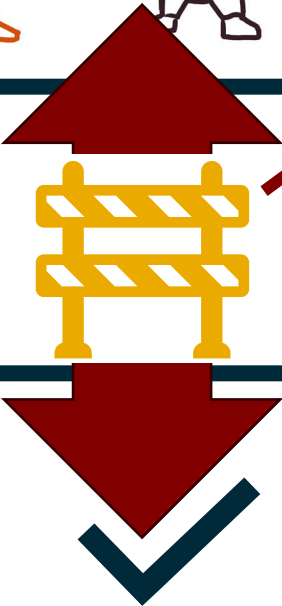
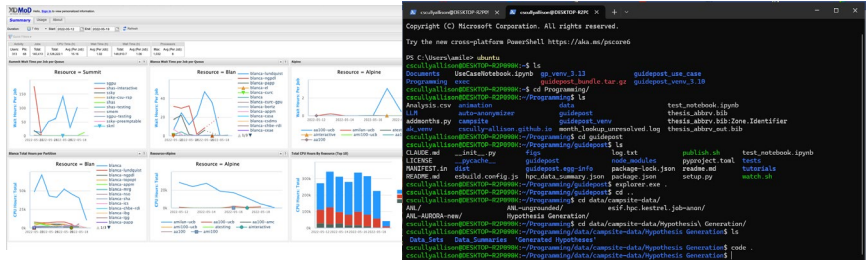
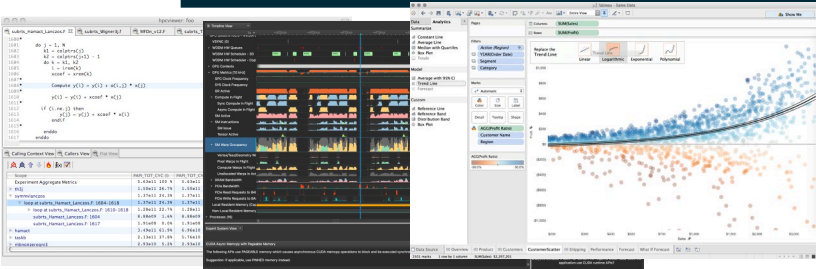
Performance Engineer



Operational Data Analyst



Tooling Gap



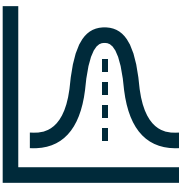
Code



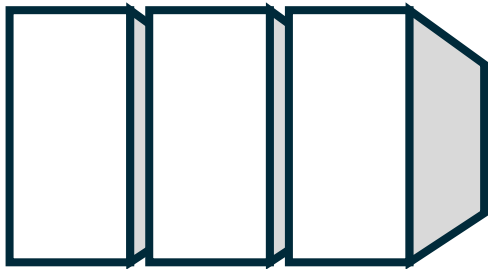
Performance Data



Verification Data



Telemetry Data

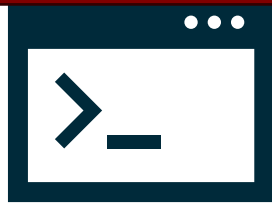


The Human

Data Gap



Why AI and not more traditional tools?



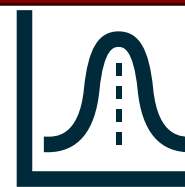
Code



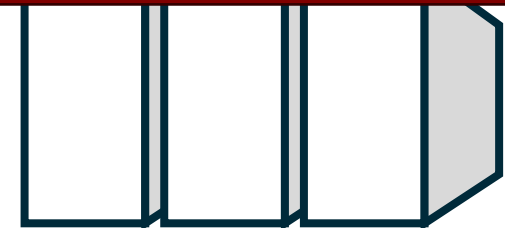
Performance Data



Verification Data



Telemetry Data



The Tooling Gap in HPC Research and Analysis

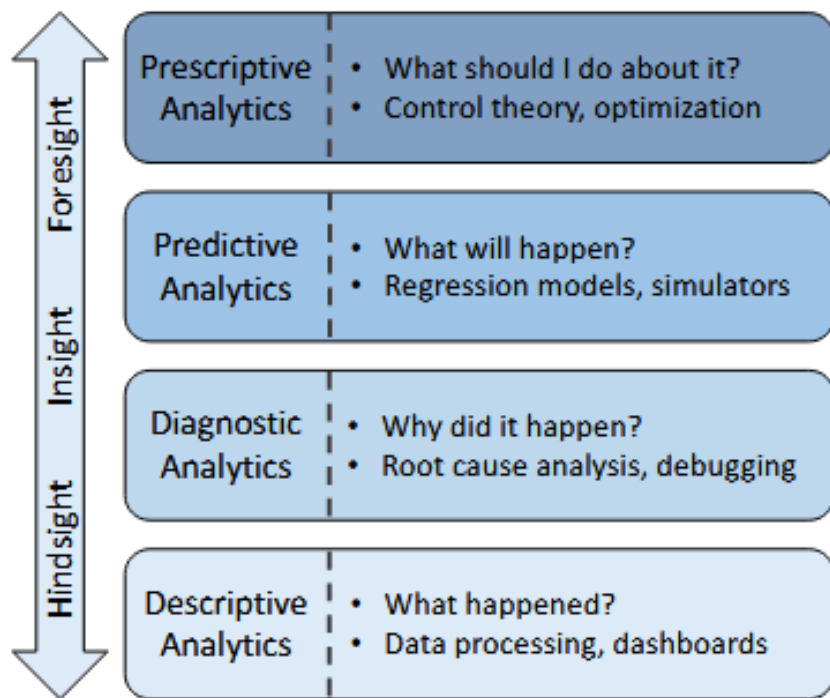


Fig. 2. The four types of data analytics [70].

[1] A. Netti, W. Shin, M. Ott, T. Wilde, and N. Bates, "A Conceptual Framework for HPC Operational Data Analytics," in *2021 IEEE International Conference on Cluster Computing (CLUSTER)*, Sep. 2021, pp. 596–603. doi: [10.1109/Cluster48925.2021.00086](https://doi.org/10.1109/Cluster48925.2021.00086).

The Tooling Gap in HPC Research and Analysis

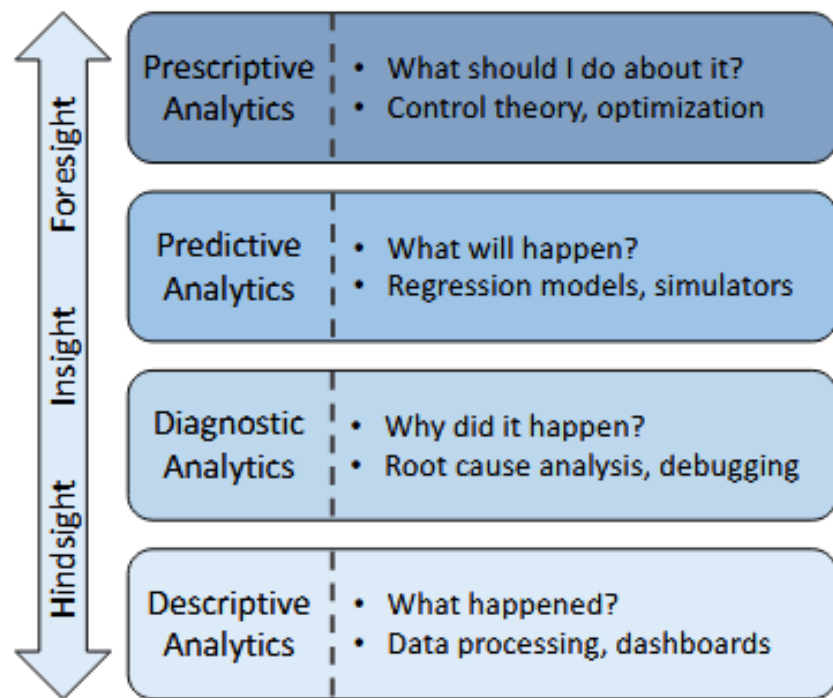
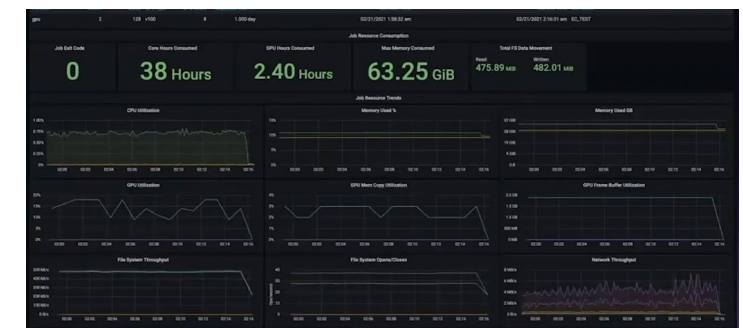
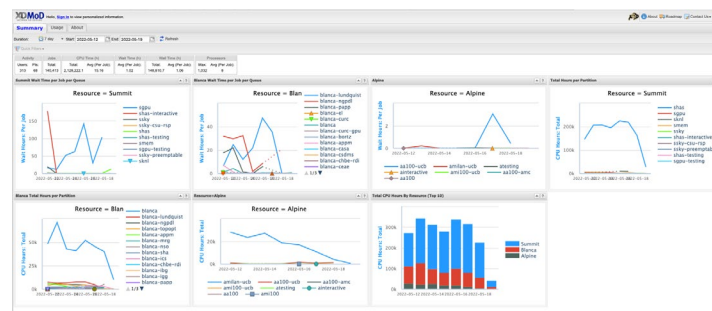
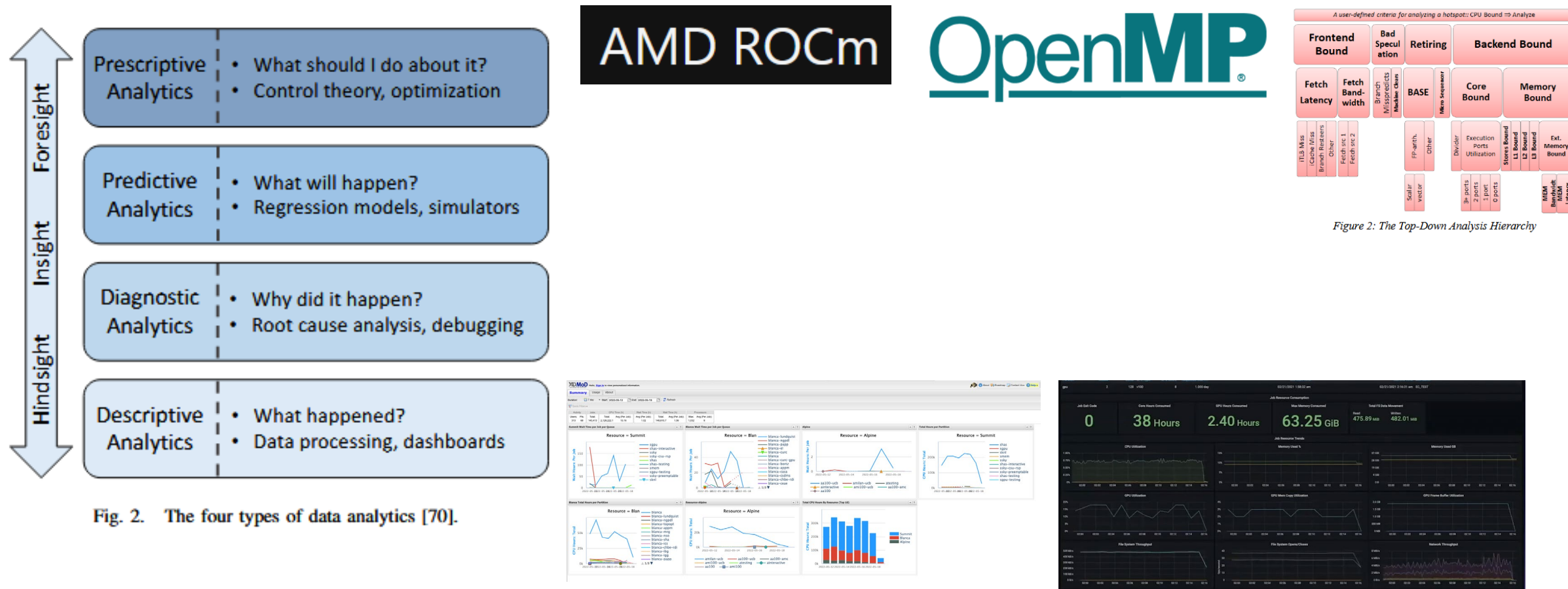


Fig. 2. The four types of data analytics [70].



[1] A. Netti, W. Shin, M. Ott, T. Wilde, and N. Bates, “A Conceptual Framework for HPC Operational Data Analytics,” in *2021 IEEE International Conference on Cluster Computing (CLUSTER)*, Sep. 2021, pp. 596–603. doi: [10.1109/Cluster48925.2021.00086](https://doi.org/10.1109/Cluster48925.2021.00086).

The Tooling Gap in HPC Research and Analysis



[1] A. Netti, W. Shin, M. Ott, T. Wilde, and N. Bates, "A Conceptual Framework for HPC Operational Data Analytics," in *2021 IEEE International Conference on Cluster Computing (CLUSTER)*, Sep. 2021, pp. 596–603. doi: [10.1109/Cluster48925.2021.00086](https://doi.org/10.1109/Cluster48925.2021.00086).

Predictive and Diagnostic Analytics are Frontiers and Require Bespoke Workflows

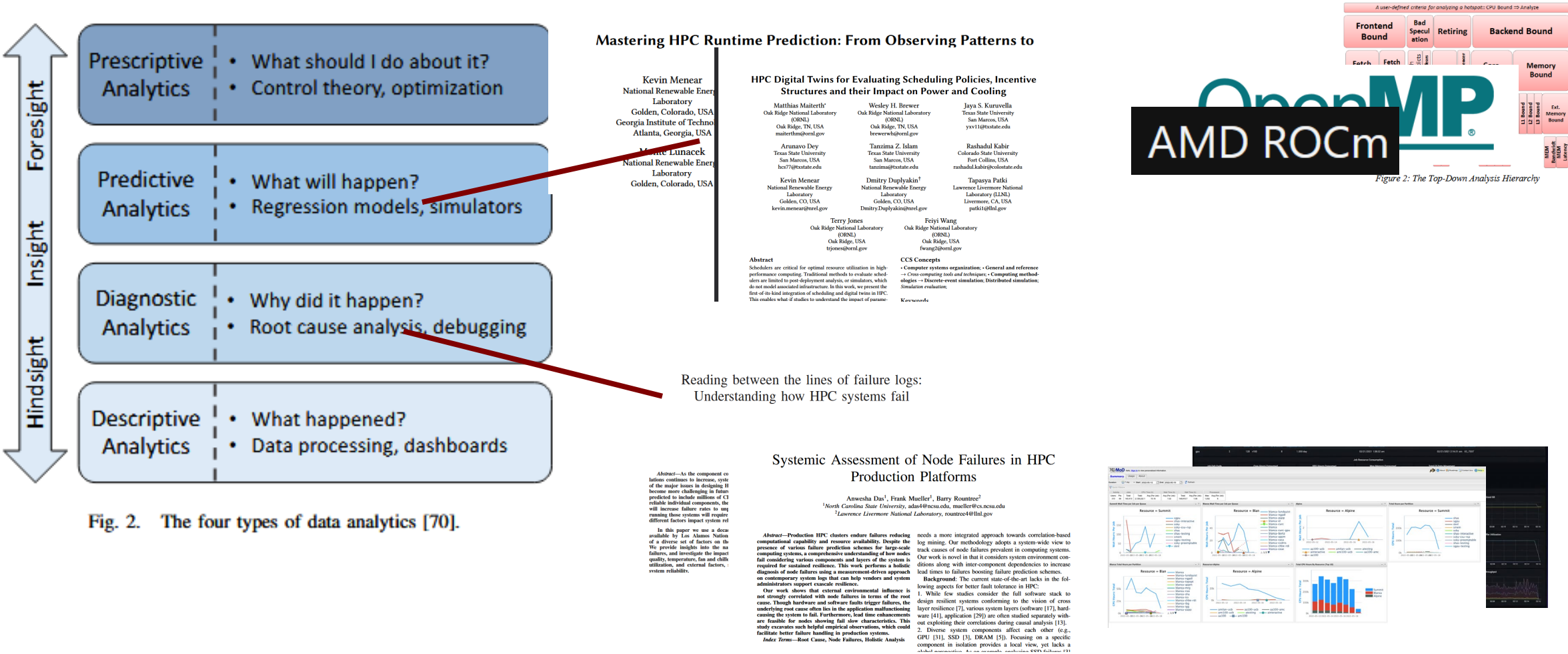


Fig. 2. The four types of data analytics [70].

[1] A. Netti, W. Shin, M. Ott, T. Wilde, and N. Bates, "A Conceptual Framework for HPC Operational Data Analytics," in 2021 IEEE International Conference on Cluster Computing (CLUSTER), Sep. 2021, pp. 596–603. doi: 10.1109/Cluster48925.2021.00086.

The Tools are Lacking for Nuanced Analysis

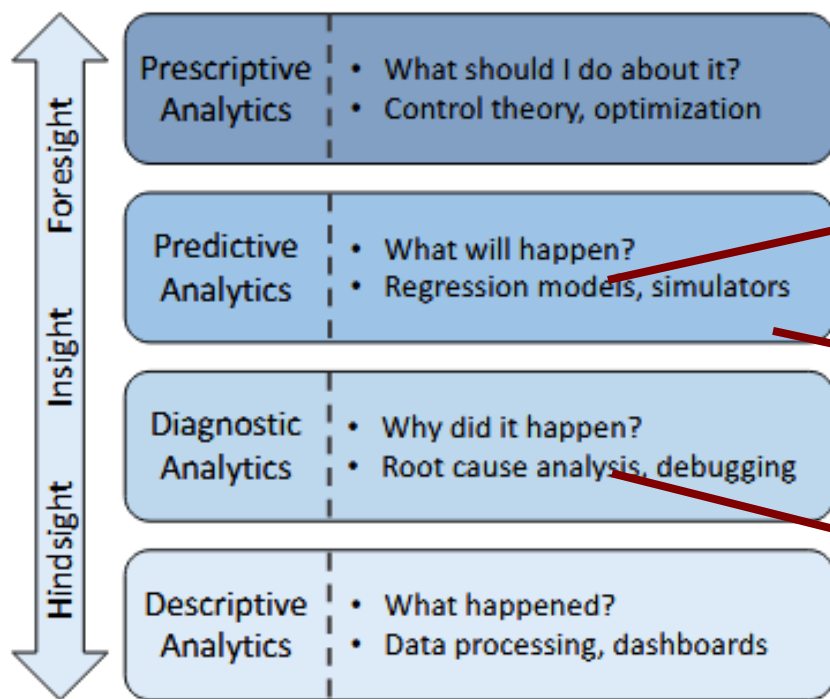


Fig. 2. The four types of data analytics [70].

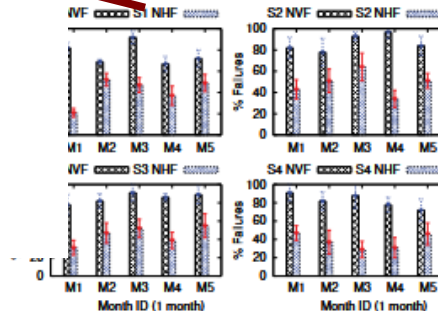
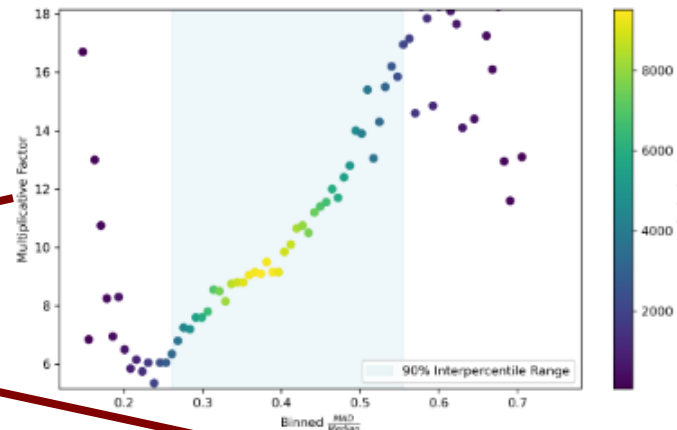


Fig. 5: Node Faults

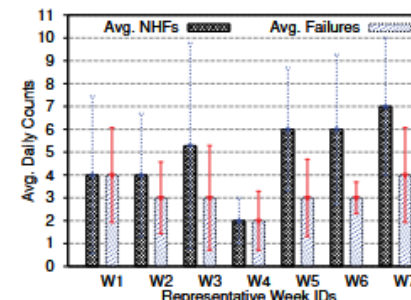
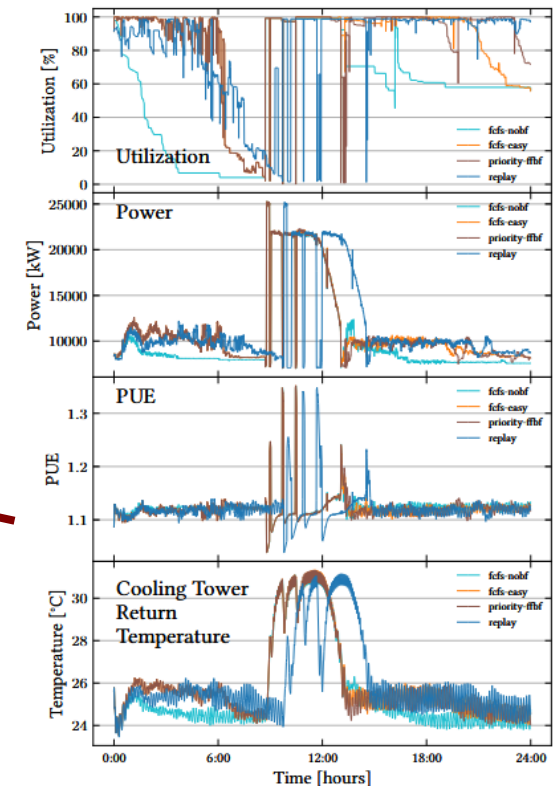


Fig. 6: Node Heartbeat Faults



[1] A. Netti, W. Shin, M. Ott, T. Wilde, and N. Bates, "A Conceptual Framework for HPC Operational Data Analytics," in *2021 IEEE International Conference on Cluster Computing (CLUSTER)*, Sep. 2021, pp. 596–603. doi: [10.1109/Cluster48925.2021.00086](https://doi.org/10.1109/Cluster48925.2021.00086).

Can LLMs Help?

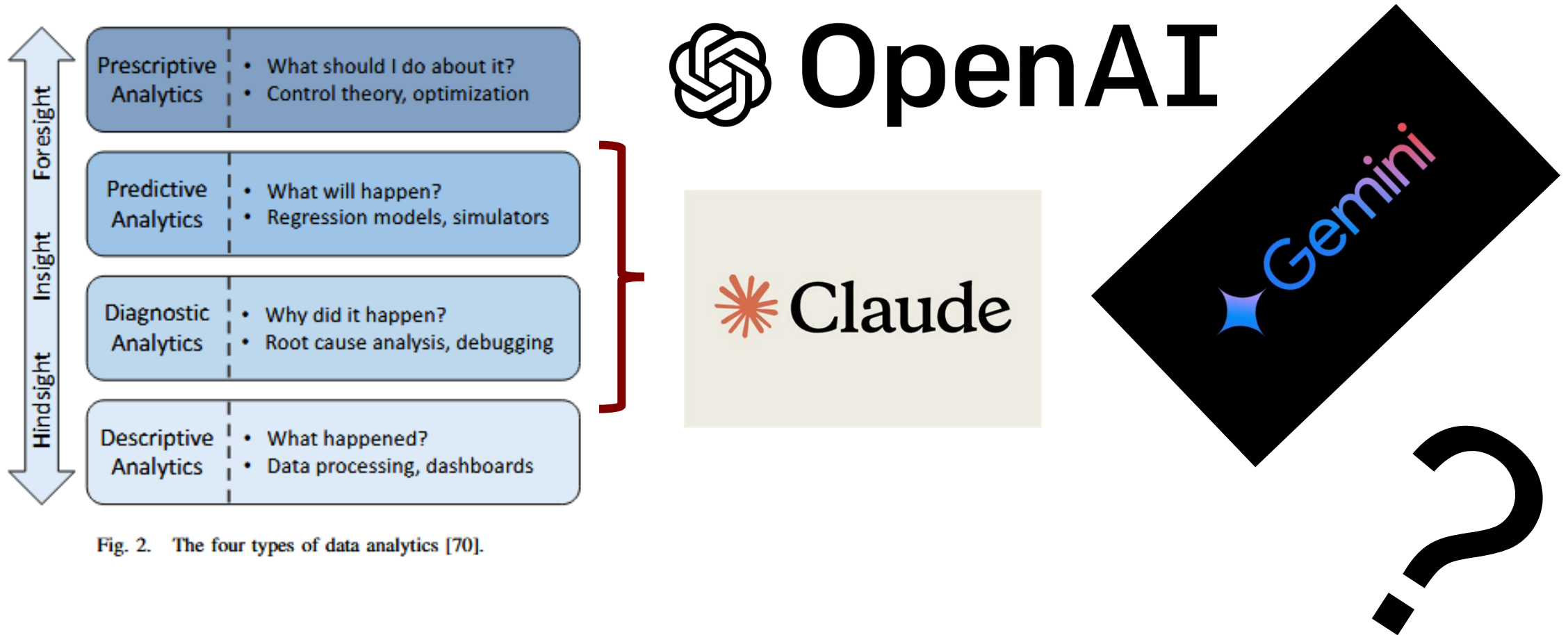


Fig. 2. The four types of data analytics [70].

[1] A. Netti, W. Shin, M. Ott, T. Wilde, and N. Bates, "A Conceptual Framework for HPC Operational Data Analytics," in *2021 IEEE International Conference on Cluster Computing (CLUSTER)*, Sep. 2021, pp. 596–603. doi: [10.1109/Cluster48925.2021.00086](https://doi.org/10.1109/Cluster48925.2021.00086).

Can LLMs Help?



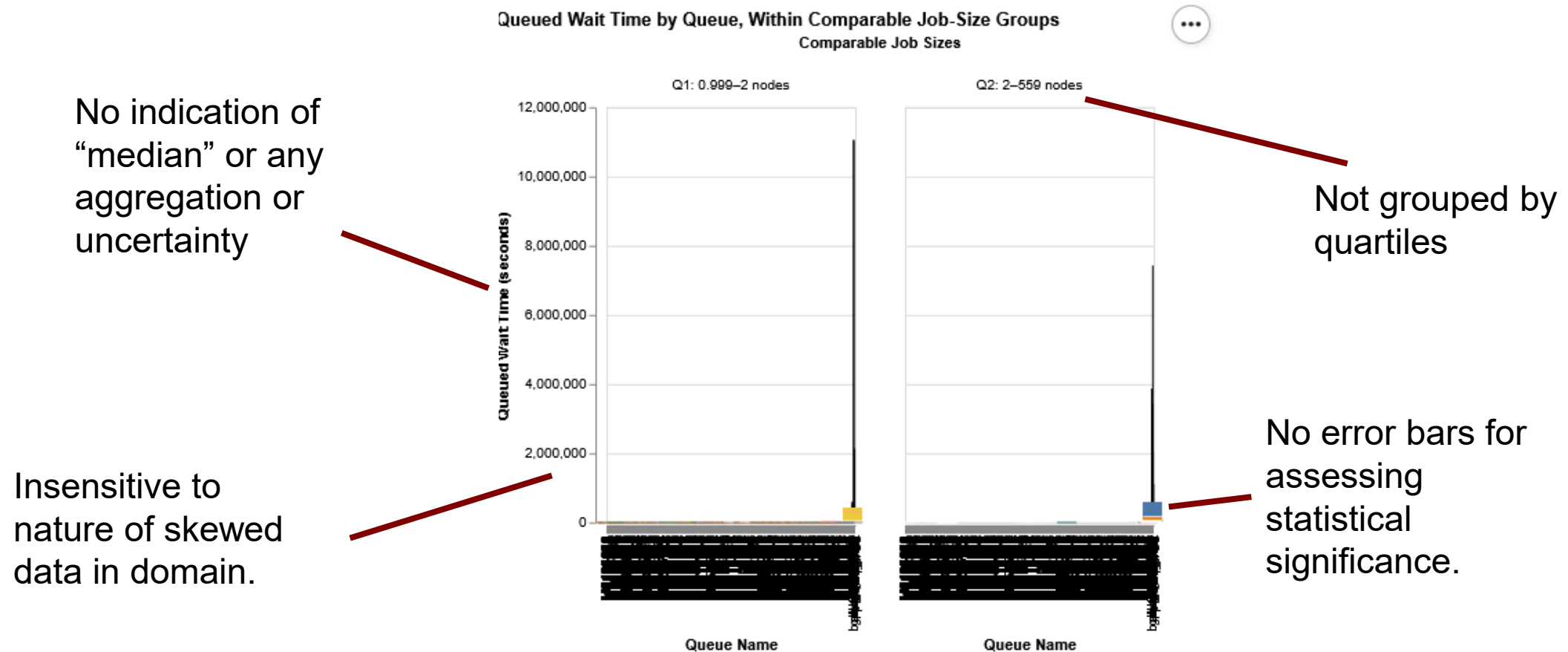
Maybe. But there are problems mitigating quick and easy adoption.

Fig. 2. The four types of data analytics [70].

[1] A. Netti, W. Shin, M. Ott, T. Wilde, and N. Bates, "A Conceptual Framework for HPC Operational Data Analytics," in *2021 IEEE International Conference on Cluster Computing (CLUSTER)*, Sep. 2021, pp. 596–603. doi: [10.1109/Cluster48925.2021.00086](https://doi.org/10.1109/Cluster48925.2021.00086).

AI still poor at complex visualization and formal hypothesis testing.*

Hypothesis: Within comparable job sizes (grouped by quartiles of NODES_REQUESTED), median QUEUED_WAIT_SECONDS differs across QUEUE_NAME.



AI weak at complex visualization and modeling?

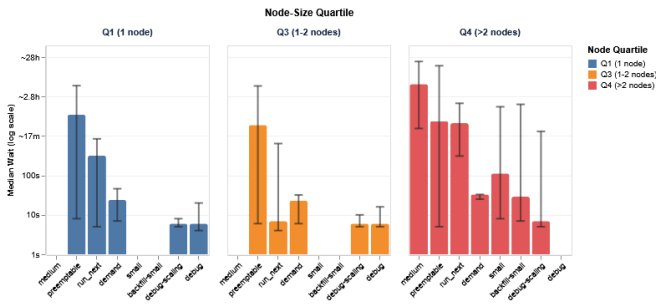
Polaris HPC — Queue Wait Time Analysis (2025)

ALCF Polaris · 262,700 jobs · Jan-Dec 2025 · Top 8 queues by volume

Hypothesis: Within comparable job sizes (grouped by quartiles of NODES_REQUESTED), median QUEUED_WAIT_SECONDS differs across QUEUE_NAME.

Chart 1 — Median Wait by Queue, Faceted by Node-Size Quartile (log scale)

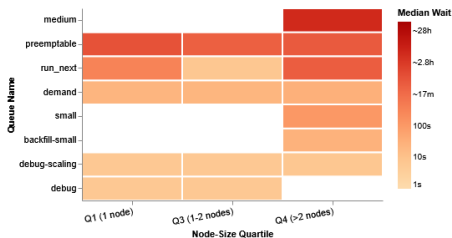
Each panel = one node-size quartile. Compare bar heights across queues within the same panel to test the hypothesis. Error bars = IQR (Q25–Q75). Hover for exact values.



Node quartiles: Q1 = 1 node · Q3 = 1–2 nodes · Q4 = >2 nodes (Q1=Q2 due to skew — 75% of all jobs request just 1 node). medium queue only appears in Q4 (all its jobs request >2 nodes).

Chart 2 — Heatmap: Median Wait across Queue × Node Quartile

Color = $\log(\text{median wait})$. Dark red = long wait. Empty cells = queue has no jobs in that size bucket.



Job Failure Clustering — ALCF Polaris 2025

Hypothesis: "Job failures tend to cluster in time rather than occurring randomly."

262,700 122,160 0.794 4.1 min

Total Jobs Failed Jobs (46.5%) Lag-1 Autocorrelation Mean Time Between Failures

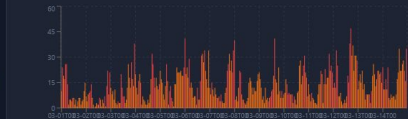
Ⓢ Daily Failure Rate — Full Year

Random process = flat line near 46.5% mean (amber dashed). Dramatic swings signal non-random, bursty behaviour.



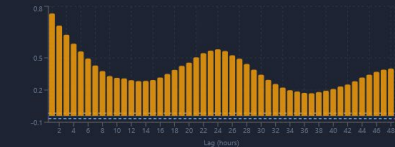
Ⓢ Hourly Failures — March 1–14 Sample

Poisson noise looks flat. Runs of tall bars then quiet stretches = clustering signature.



Ⓢ Autocorrelation of Hourly Failure Counts (lags 1–48 h)

Blue lines = 95% CI under Poisson (±0.021). All 48 bars exceed 0 — strong serial correlation.



Ⓢ Inter-Failure Time Distribution vs. Exponential Baseline

Pure Poisson = exponential gaps (amber lines). Observed (red) piles up at near-zero and has a heavy tail — classic burst signature.



Verdict

- Ⓢ Daily failure rate swings from near 0% to ~86% — far from a flat random baseline.
- Ⓢ Hour-level data shows clear burst episodes — extended high-failure windows, then quiet periods.
- Ⓢ Lag-1 ACF = 0.794 (Poisson baseline = 0). Every single lag from 1–48h exceeds the 95% CI, showing deep serial dependence.
- Ⓢ Inter-failure times are far more front-loaded (many near-simultaneous failures) and have a heavier tail than the exponential — the textbook clustering signature.

All four tests strongly support the hypothesis: job failures on Polaris cluster in time and are not random.

CHART 1 — TEMPORAL CLUSTERING

Daily failure rate across 2025

If failures were random (Poisson), the daily rate would hover near a stable mean. Visible bursts suggest causal episodes.

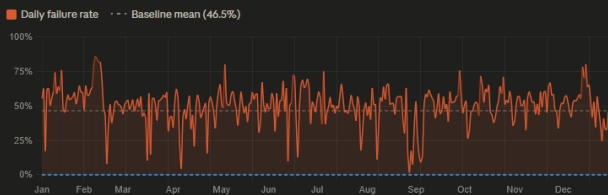


CHART 2 — AUTOCORRELATION (ACF)

Autocorrelation of daily failure counts

Under a random (Poisson) null, all bars should fall within the blue confidence band (±1.96/√n). Values outside the band confirm memory in the failure process.

ACF at lag k 95% confidence interval

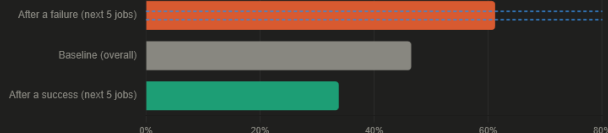


CHART 3 — CONDITIONAL FAILURE RATE

Failure rate of next 5 jobs, conditioned on preceding outcome

If failures are random, the outcome of the preceding job should not predict the next job's failure rate. The large gap here is direct evidence of causal clustering.

After a failure (61.2%) After a success (33.8%) Baseline (46.5%)



✓ **Hypothesis supported.** Three independent tests all point the same way: (1) The time series shows prolonged high-failure episodes inconsistent with a stable-rate Poisson process. (2) The ACF at lag 1 is 0.70 — far beyond the 95% CI of ±0.10 — meaning today's failure count strongly predicts tomorrow's. (3) The conditional failure rate after a failure (61%) is nearly double that after a success (34%), an 81% relative lift

Issues even with good generation.

Manual data uploads to proprietary software platform.

I am going to give you a dataset. Can you please produce an altair visualization that allows me to test the following hypothesis 'Within comparable job sizes (grouped by quartiles of NODES_REQUESTED), median QUEUED_WAIT_SECONDS differs across QUEUE_NAME.' against this dataset.

Please share the dataset and I'll build the Altair visualization for you!

ANL-ALCF-DJC-POLARIS_20250101_20251231.csv

GZ

Ran 6 commands, viewed a file >

Now I have everything I need. Let me build a rich Altair visualization as an HTML artifact:

Created a file, read a file >

The visualization has two complementary panels:

Chart 1 — Grouped bar chart with IQR whiskers (log scale): Each queue is a group, bars are colored by node-size quartile, and the vertical rules show the interquartile range of wait times. This directly exposes whether median wait shifts within the same quartile as you move across queues.

Chart 2 — Heatmap (log color scale): Rows = queues, columns = node quartiles, color intensity = median wait. This makes cross-queue comparisons within each column (same job size) immediately visible.

Key findings supporting the hypothesis:

- Within Q4 (>2 nodes), `medium` has a median wait of ~20,861 s, while `debug-scaling`, `backfill-small`, and `demand` sit at 7-33 s — a ~3,000× difference.
- Within Q1 (<1 node), `preemptable` waits ~3,489 s vs. `debug / debug-scaling` at just 6 s.
- The hypothesis is strongly supported: queue identity matters far more than job size in determining wait time on Polaris.

Polaris wait viz

Code - HTML

Download

Download and manual integration into notebook and analysis workflows necessary.

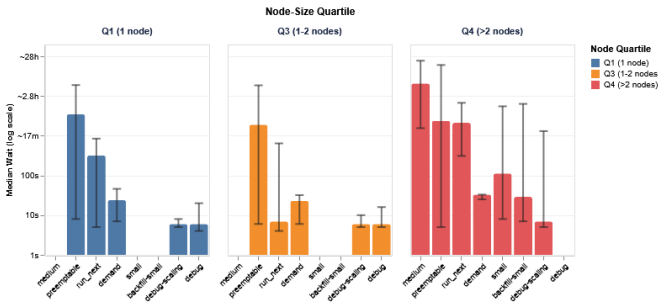
Polaris HPC — Queue Wait Time Analysis (2025)

ALCF Polaris - 262,700 jobs - Jan-Dec 2025 - Top 8 queues by volume

Hypothesis: Within comparable job sizes (grouped by quartiles of NODES_REQUESTED), median QUEUED_WAIT_SECONDS differs across

Chart 1 — Median Wait by Queue, Faceted by Node-Size Quartile (log scale)

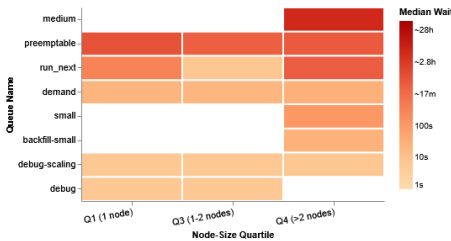
Each panel = one node-size quartile. Compare bar heights across queues within the same panel to test the hypothesis. Error bars = IQR (Q25-Q75). How



Y-axis shows $\log_{10}(\text{median wait in seconds})$. Tick labels are back-transformed to seconds for readability. Node quartiles: Q1 = 1 node · Q3 = 1-2 nodes · Q4 = >2 nodes (Q1=Q2 due to skew — 75% of all jobs request just 1 node). `medium` queue only appears in Q4 (all its jobs request >2 nodes).

Chart 2 — Heatmap: Median Wait across Queue × Node Quartile

Color = $\log(\text{median wait})$. Dark red = long wait. Empty cells = queue has no jobs in that size bucket.



No guarantees of accurate translation between natural language and visual representation.

No provenance on production of visualizations and report.

Issues even with good generation.

Manual data uploads
to proprietary
software platform.

I am going to give you a dataset. Can you please produce an altair visualization that allows me to test the following hypothesis 'Within comparable job sizes (grouped by quartiles of NODES_REQUESTED), median QUEUED_WAIT_SECONDS differs across QUEUE_NAME.' against this dataset.

Please share the dataset and I'll build the Altair visualization for you!

ANL-ALCF-DJC-POLARIS_20250101_20251231.csv

GZ

Ran 6 commands, viewed a file >

Now I have everything I need. Let me build a rich Altair visualization as an HTML artifact:

Created a file, read a file >

The visualization has two complementary panels:

Chart 1 — Grouped bar chart with IQR whiskers (log scale): Each queue is a group, bars are colored by node-size quartile, and the vertical rules show the interquartile range of wait times. This directly exposes whether median wait shifts within the same quartile as you move across queues.

Chart 2 — Heatmap (log color scale): Rows = queues, columns = node quartiles, color intensity = median wait. This makes cross-queue comparisons within each column (same job size) immediately visible.

Key findings supporting the hypothesis:

- Within Q4 (>2 nodes), `medium` has a median wait of ~20,861 s, while `debug-scaling`, `backfill-small`, and `demand` sit at 7-33 s — a ~3,000× difference.
- Within Q1 (<1 node), `preemptable` waits ~3,489 s vs. `debug / debug-scaling` at just 6 s.
- The hypothesis is strongly supported: queue identity matters far more than job size in determining wait time on Polaris.

Polaris wait viz

Code - HTML

Download

Download and
manual integration
into notebook and
analysis workflows
necessary.

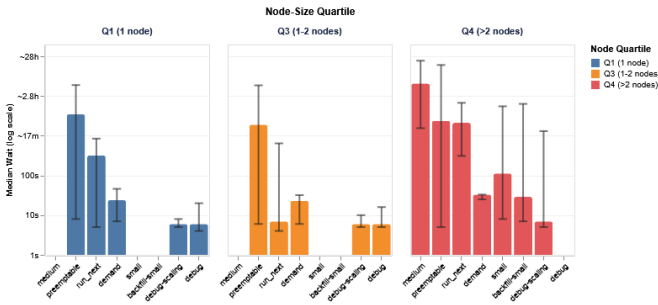
Polaris HPC — Queue Wait Time Analysis (2025)

ALCF Polaris - 262,700 jobs - Jan-Dec 2025 - Top 8 queues by volume

Hypothesis: Within comparable job sizes (grouped by quartiles of NODES_REQUESTED), median QUEUED_WAIT_SECONDS differs across

Chart 1 — Median Wait by Queue, Faceted by Node-Size Quartile (log scale)

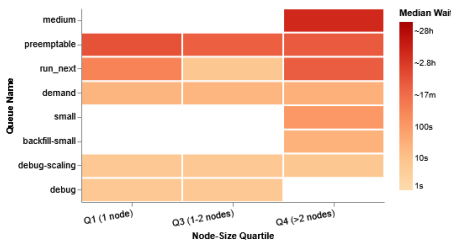
Each panel = one node-size quartile. Compare bar heights across queues within the same panel to test the hypothesis. Error bars = IQR (Q25-Q75). How



Y-axis shows $\log_{10}$ (median wait in seconds). Tick labels are back-transformed to seconds for readability.
Node quartiles: Q1 = 1 node · Q3 = 1-2 nodes · Q4 = >2 nodes (Q1=Q2 due to skew — 75% of all jobs request just 1 node).
`medium` queue only appears in Q4 (all its jobs request >2 nodes).

Chart 2 — Heatmap: Median Wait across Queue × Node Quartile

Color = $\log$ (median wait). Dark red = long wait. Empty cells = queue has no jobs in that size bucket.



No guarantees of
accurate translation
between natural
language and visual
representation.

No provenance on production of
visualizations and report.

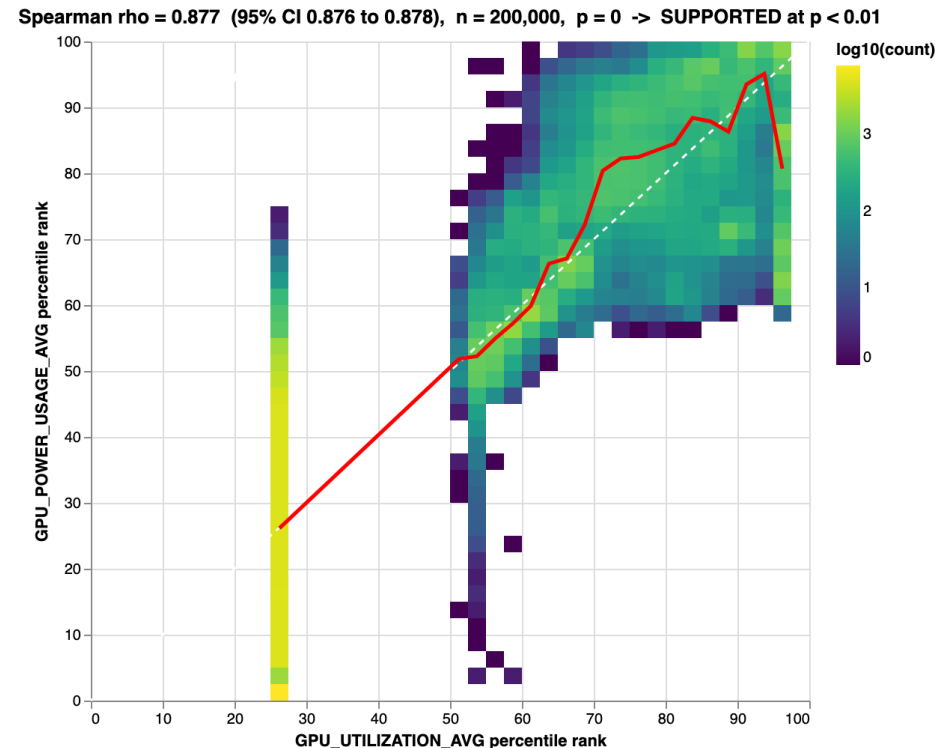


Part 1: Getting Good VIS from LLMs

Translation Fidelity

Natural Language: “Is there a positive correlation between my system’s GPU utilization and my system’s power usage?”

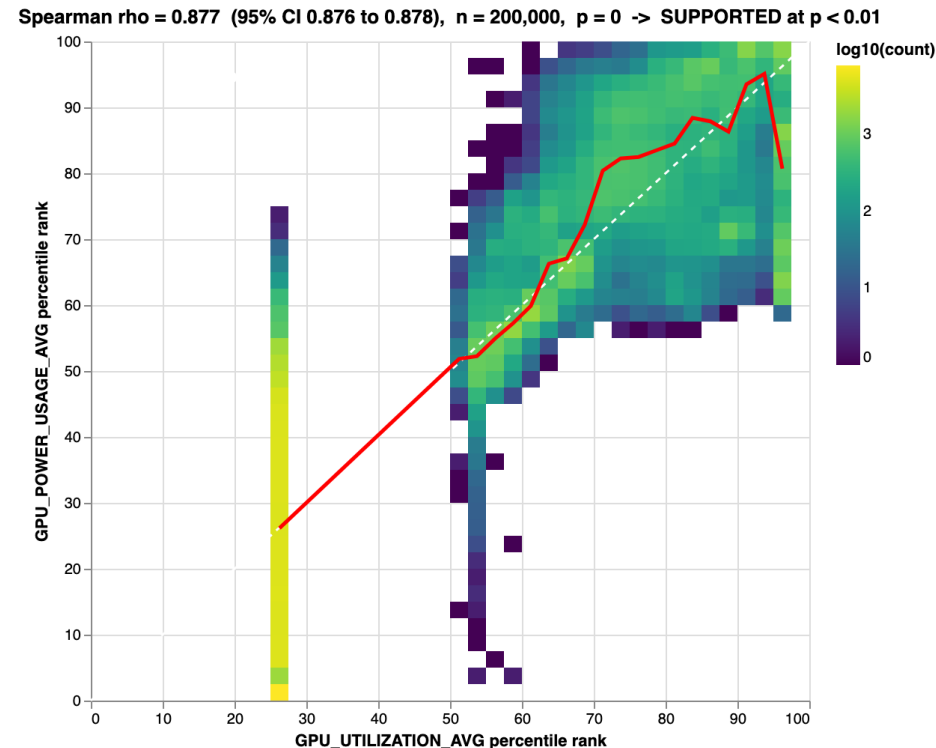
Visualization:



Translation Fidelity

Natural Language: “Is there a positive correlation between my system’s GPU utilization and my system’s power usage?”

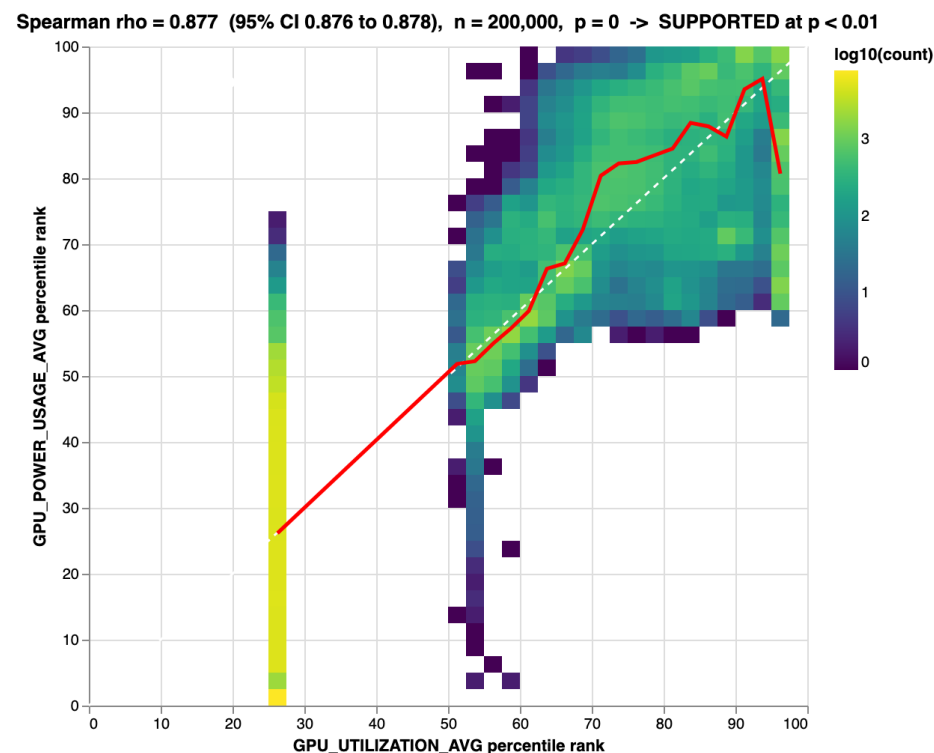
Visualization:



Translation Fidelity

Natural Language: “Is there a positive correlation between my system’s GPU utilization and my system’s power usage?”

Visualization:



Red Line?

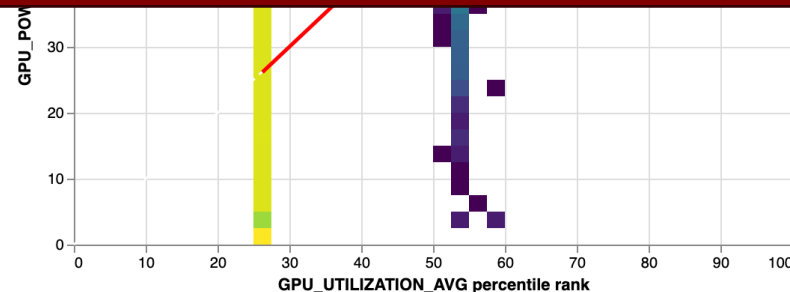
Dotted Line?



Translation Fidelity

Natural Language: “Is there a positive correlation between my system’s GPU utilization and my system’s power usage?”

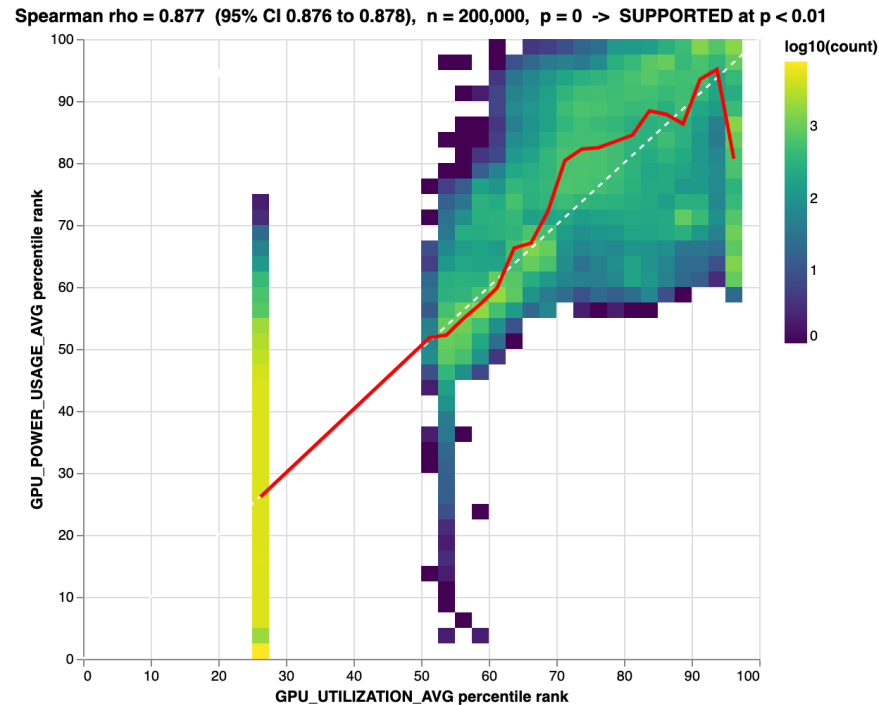
How can we measure this drift systematically and fix this issue?



Semantic Invariants

Natural Language: “Is there a positive correlation between my system’s GPU utilization and my system’s power usage?”

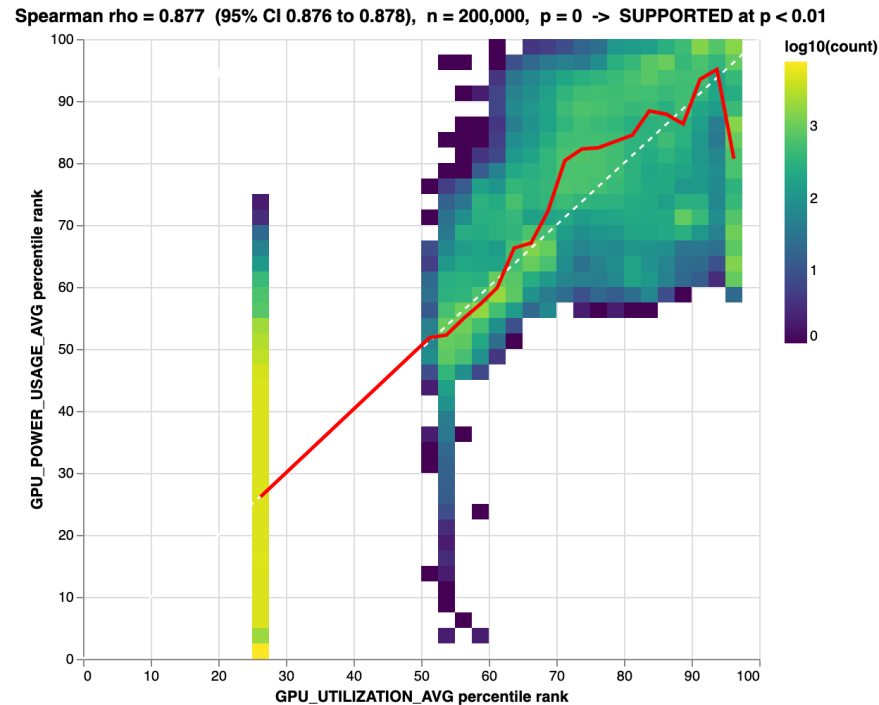
Visualization:



Semantic Invariant Example – Quantity of Interest

Natural Language: “Is there a positive correlation between my system’s GPU utilization and my system’s power usage?”

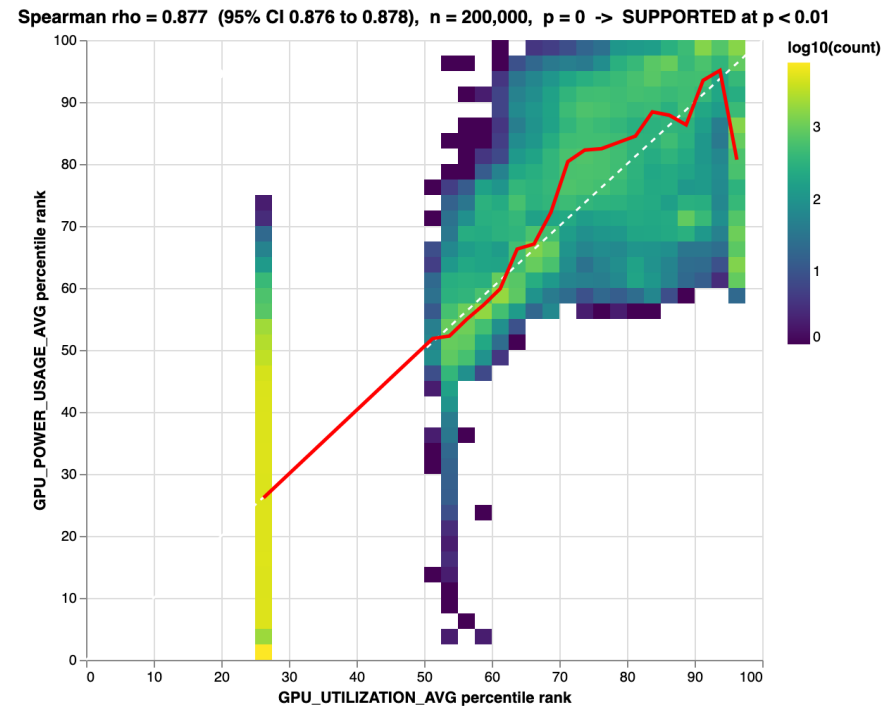
Visualization:



Semantic Invariant Example – Quantity of Interest

Natural Language: “Is there a positive correlation between my system’s GPU utilization and my system’s power usage?”

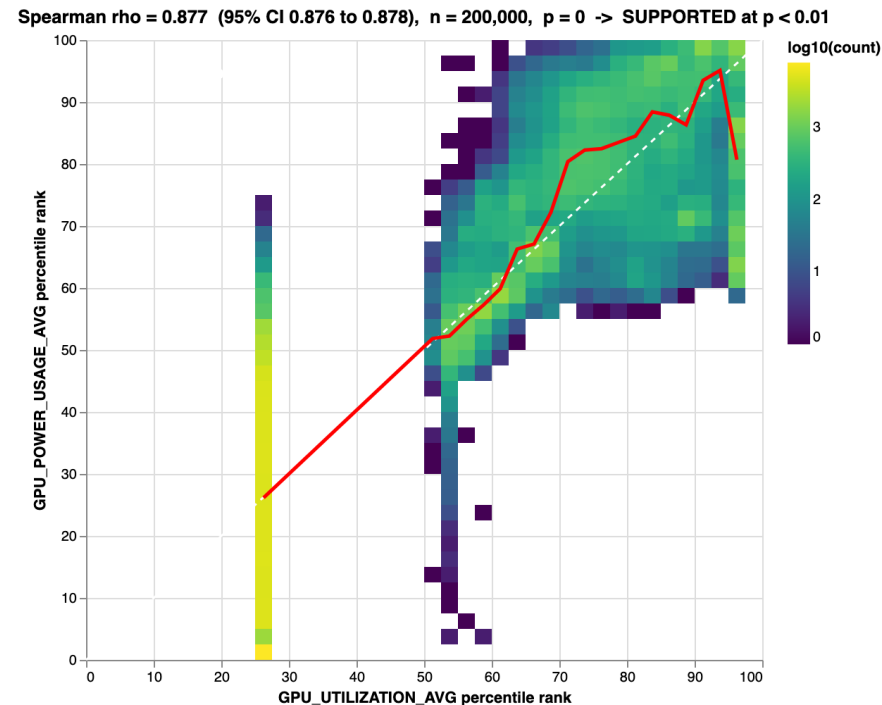
Visualization:



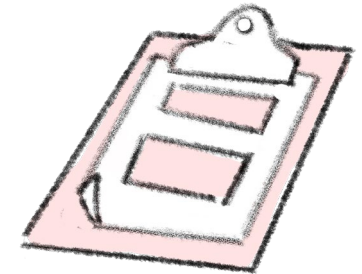
Semantic Invariant Example – Quantity of Interest

Natural Language: “Is there a positive correlation between my system’s GPU utilization and my system’s power usage?”

Visualization:



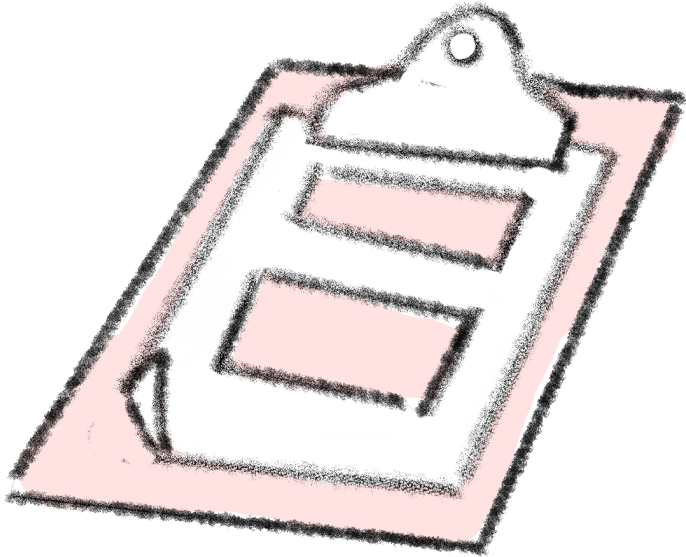
Semantic Invariants



A checklist of semantics expressed by the natural language translation that should not change during the translation process.

Semantic Invariants

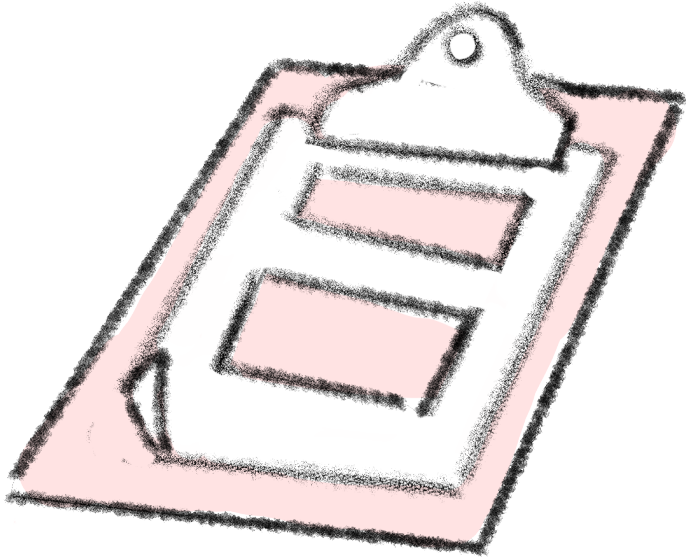
Natural Language: “Is there a positive correlation between my system’s GPU utilization and my system’s power usage?”



- Is there modeling or uncertainty in the hypothesis?
(uncertainty, modeling)
- What is the hypothesis about?
(quantity, comparator, referent)
- Are there any filters or groups defined over the data?
(conditioning)

Semantic Invariants

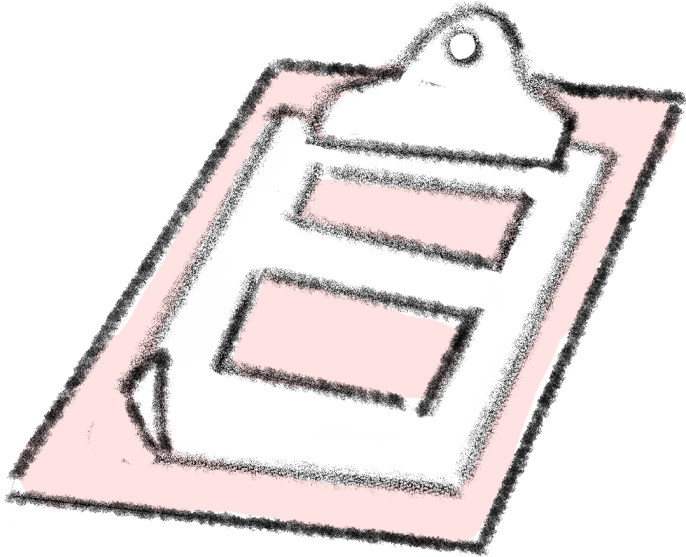
Natural Language: “Is there a positive correlation between my system’s GPU utilization and my system’s power usage?”



- Is there modeling or uncertainty in the hypothesis?
(uncertainty, modeling)
- What is the hypothesis about?
(quantity, comparator, referent)
- Are there any filters or groups defined over the data?
(conditioning)

Semantic Invariants

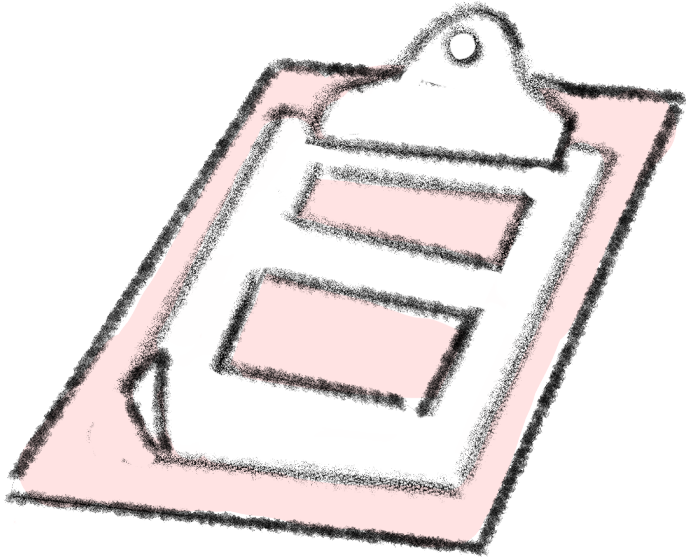
Natural Language: “Is there a positive correlation between my system’s GPU utilization and my system’s power usage?”



- Is there modeling or uncertainty in the hypothesis?
(uncertainty, modeling)
- What is the hypothesis about?
(quantity, comparator, referent)
- Are there any filters or groups defined over the data?
(conditioning)

Semantic Invariants

Natural Language: “Is there a positive correlation between my system’s GPU utilization and my system’s power usage?”



- Is there modeling or uncertainty in the hypothesis?
(uncertainty, modeling)
- What is the hypothesis about?
(quantity, comparator, referent)
- Are there any filters or groups defined over the data?
(conditioning)

Semantic Invariants

Natural Language: “Is there a positive correlation between my system’s GPU utilization and my system’s power usage?”

- Is there modeling or

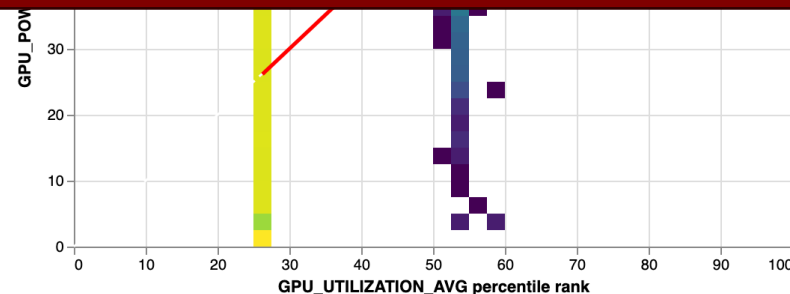
Provides us with a model to evaluate that critical **human** intent is preserved when working with LLMs for complex tasks.

(conditioning)

Translation Fidelity

Natural Language: “Is there a positive correlation between my system’s GPU utilization and my system’s power usage?”

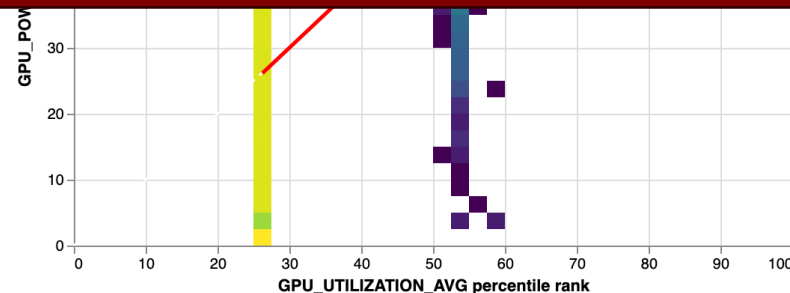
How can we measure this drift systematically and fix this issue?



Translation Fidelity

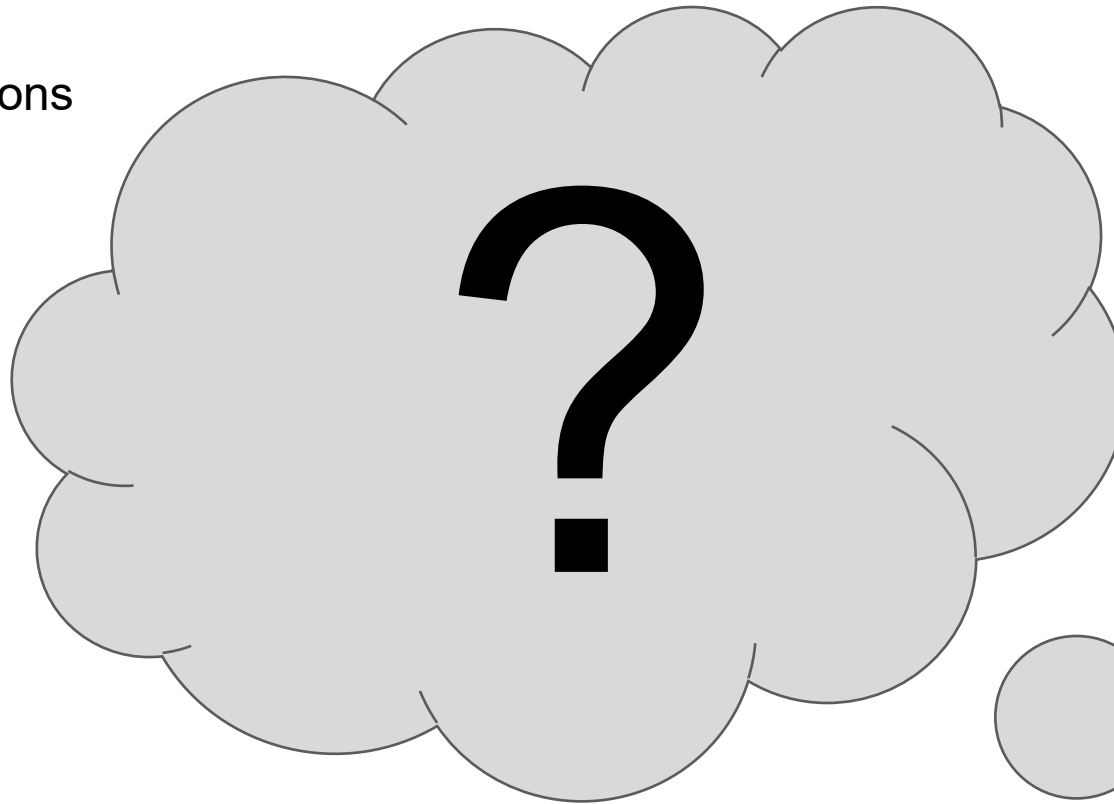
Natural Language: “Is there a positive correlation between my system’s GPU utilization and my system’s power usage?”

How can we measure this drift systematically
and fix this issue?



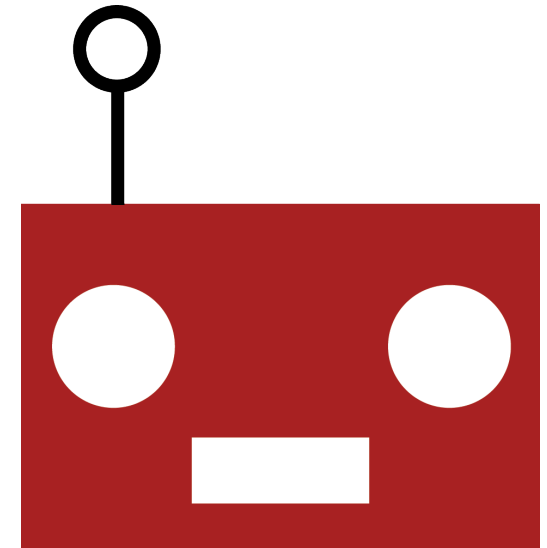
Prompt Engineering – Highly Specific and Unscientific

“Were those interventions effective?”



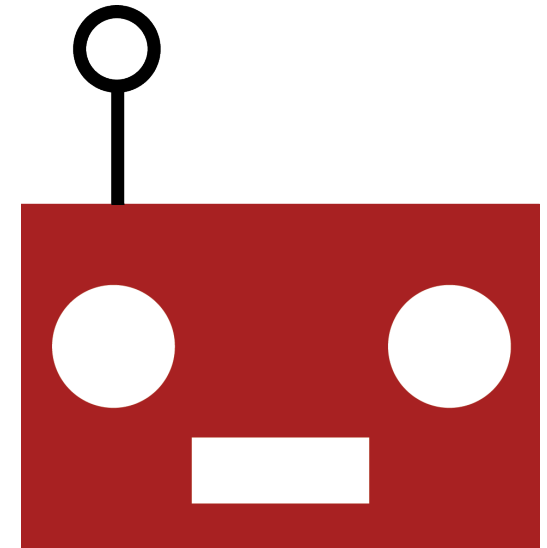
“Be absolutely, completely, certain NOT to use these visual marks.”

“You are an expert in statistical analysis and HPC.”



Using the power of formal grammars to scaffold LLM “thought.”

```
#####  
# Hypothesis spine (all versions)  
#####  
  
hypothesis  :- event  
|           | event WITHIN partition  
|           | event ACROSS partition  
|           | event ACROSS partition WITHIN partition  
  
event       :- quantity comparator referent  
|           | quantity ▷ referent  
|           | quantity comparator ▷  
  
referent    :- const | func(var (, var)*)  
  
comparator  :- > | < | = | >= | <= | != | BETWEEN | IN
```



Defining a hypothesis using a formal grammar.

```
#####  
# Hypothesis spine (all versions)  
#####  
  
hypothesis  :- event  
|           | event WITHIN partition  
|           | event ACROSS partition  
|           | event ACROSS partition WITHIN partition  
  
event       :- quantity comparator referent  
|           | quantity ∅ referent  
|           | quantity comparator ∅  
  
referent    :- const | func(var (, var)*)  
  
comparator  :- > | < | = | >= | <= | != | BETWEEN | IN
```

All formal hypotheses are claims about an **event** that data captures.

An **event** describes a relationship between a quantity of interest and reference value?

An example in the HPC space:

Natural Language: “Is there a **positive correlation** between my system’s **GPU utilization** and my system’s **power usage**?”

Using the power of formal grammars to scaffold LLM “thought.”

Natural Language: “Is there a positive correlation between my system’s GPU utilization and my system’s power usage?”

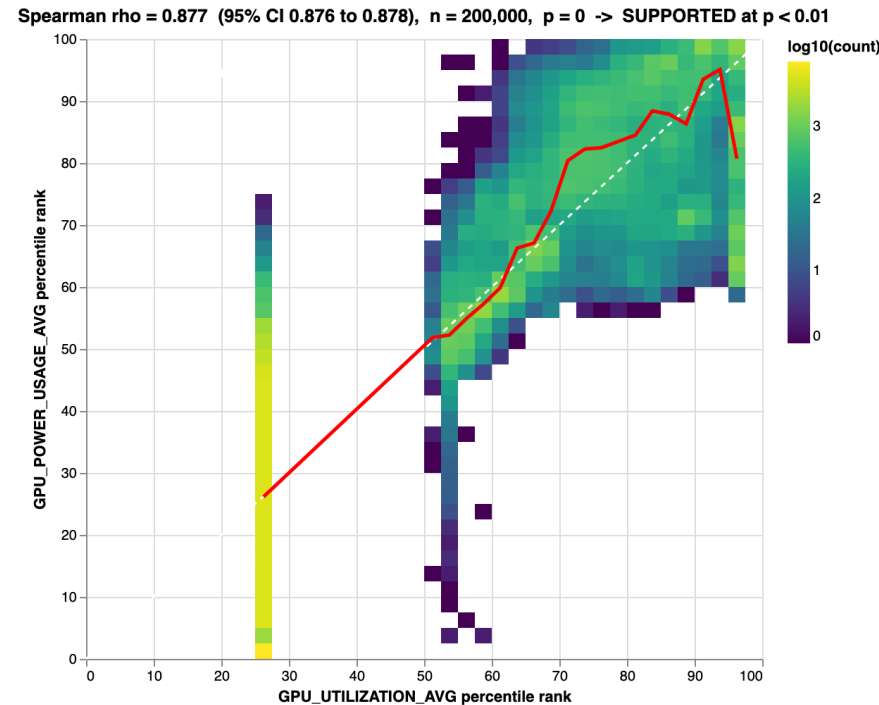


```
LET  
  
EXPR @e.rho := CORR(  
    GPU_UTILIZATION_AVG,  
    GPU_POWER_USAGE_AVG  
);  
  
PREDICATE @p.sig := p_value(  
    spearman_test(  
        [GPU_UTILIZATION_AVG],  
        [GPU_POWER_USAGE_AVG]  
    )  
) < 0.01  
  
IN  
  
@e.rho WHERE @p.sig > 0
```

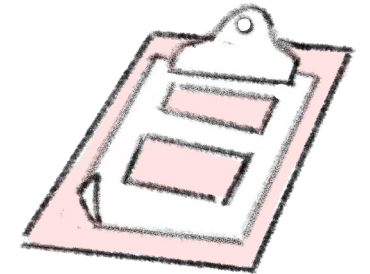
Semantic Invariants

Natural Language: “Is there a positive correlation between my system’s GPU utilization and my system’s power usage?”

Visualization:



Semantic Invariants



A checklist of semantics expressed by the natural language translation that should not change during the translation process.

Translation Fidelity

Natural Language: “Is there a positive correlation between my system’s GPU utilization and my system’s power usage?”

IR: LET

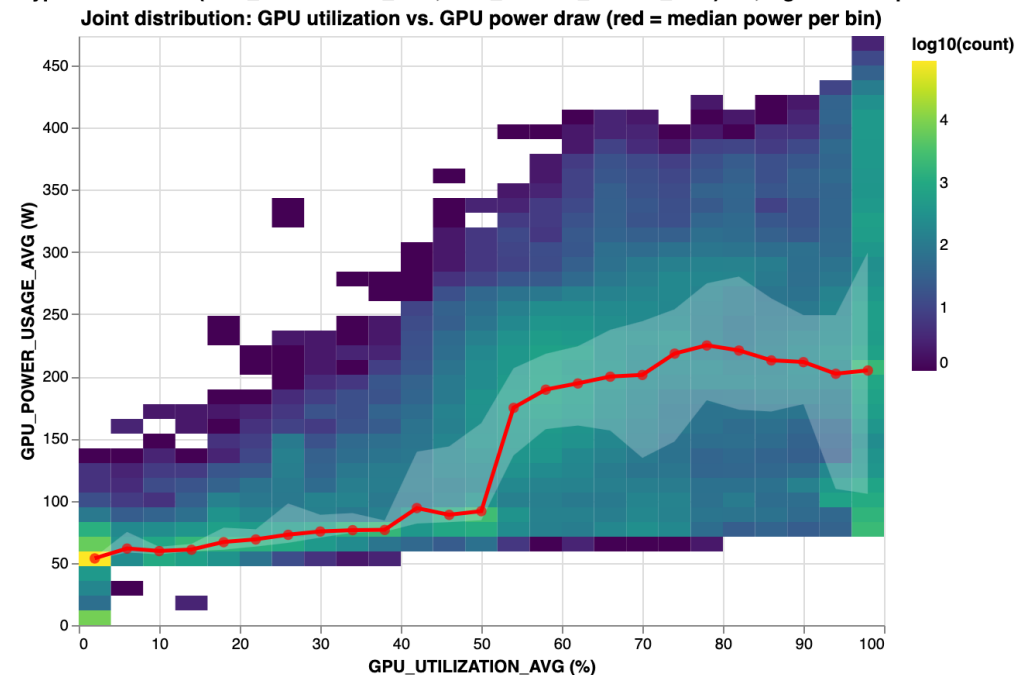
```
EXPR @e.rho := CORR(  
  GPU_UTILIZATION_AVG,  
  GPU_POWER_USAGE_AVG  
);  
  
PREDICATE @p.sig := p_value(  
  spearman_test(  
    [GPU_UTILIZATION_AVG],  
    [GPU_POWER_USAGE_AVG]  
  )  
) < 0.01
```

IN

@e.rho WHERE @p.sig > 0

Visualization:

Hypothesis: $\text{CORR}(\text{GPU_UTILIZATION_AVG}, \text{GPU_POWER_USAGE_AVG}) > 0$, significant at $p < 0.01$



Spearman test of @e.rho with @p.sig ($p < 0.01$)

Spearman rho (CORR of ranks)	0.8766
Pearson rho	0.7997
Spearman test p-value	$< 1e-300$
Significant at alpha = 0.01?	YES
rho > 0 (positive)?	YES
Hypothesis supported?	YES
n (sampled rows)	200,000

Translation Fidelity – Grammar Guided

Natural Language: “Is there a positive correlation between my system’s GPU utilization and my system’s power usage?”

IR: LET

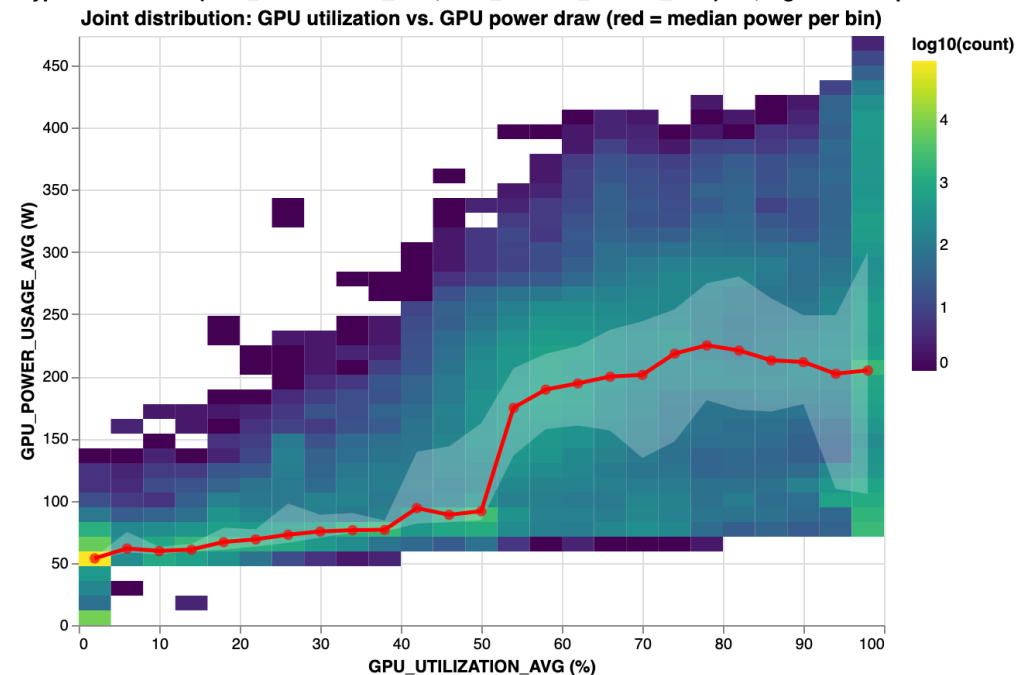
```
EXPR @e.rho := CORR(  
  GPU_UTILIZATION_AVG,  
  GPU_POWER_USAGE_AVG  
);  
  
PREDICATE @p.sig := p_value(  
  spearman_test(  
    [GPU_UTILIZATION_AVG],  
    [GPU_POWER_USAGE_AVG]  
  )  
) < 0.01
```

IN

@e.rho WHERE @p.sig > 0

Visualization:

Hypothesis: $\text{CORR}(\text{GPU_UTILIZATION_AVG}, \text{GPU_POWER_USAGE_AVG}) > 0$, significant at $p < 0.01$

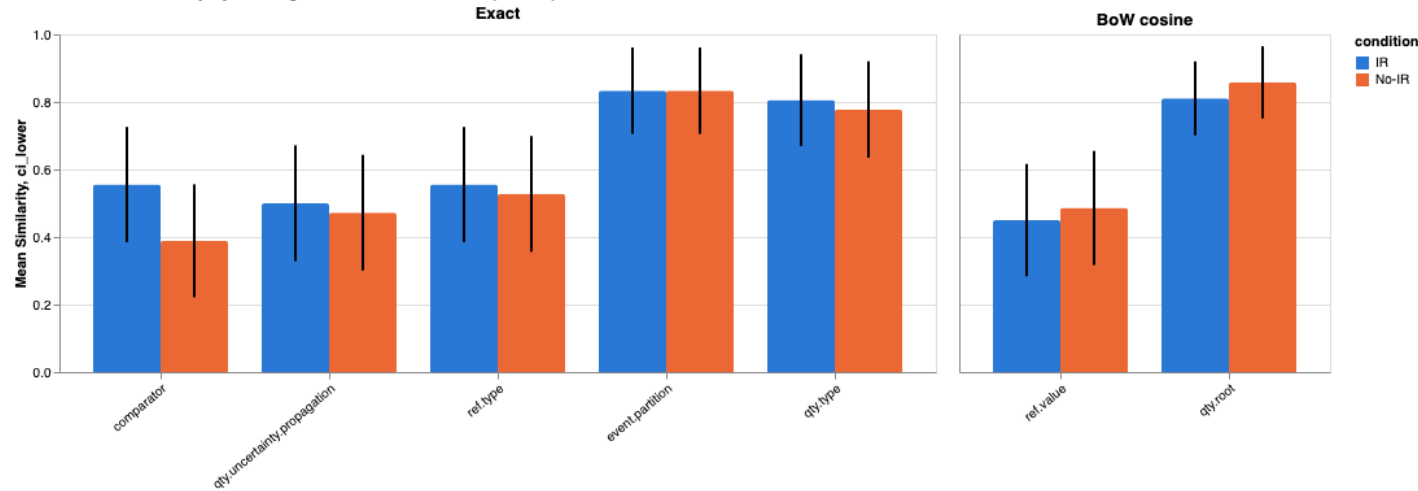


Spearman test of @e.rho with @p.sig ($p < 0.01$)

Spearman rho (CORR of ranks)	0.8766
Pearson rho	0.7997
Spearman test p-value	$< 1e-300$
Significant at alpha = 0.01?	YES
rho > 0 (positive)?	YES
Hypothesis supported?	YES
n (sampled rows)	200,000

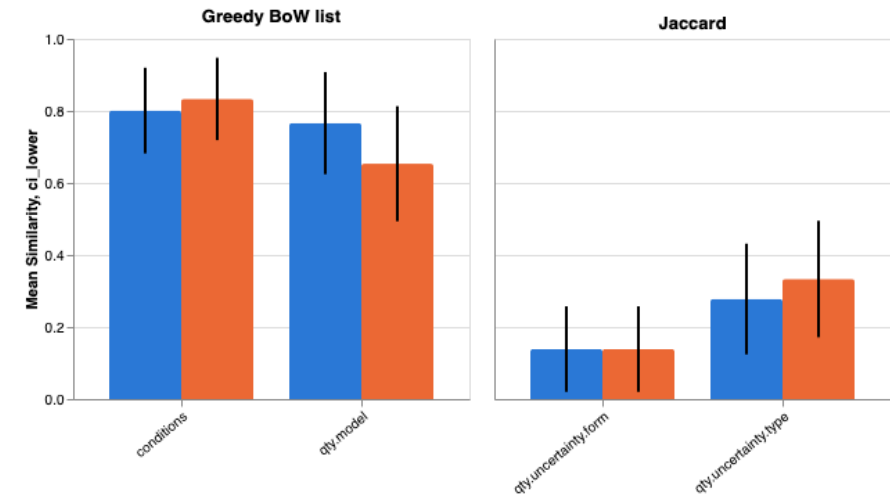
We can use semantic invariants to measure drift across many trials.

A5 — NL→VIS Similarity by Scoring Method and Condition (95% CI)



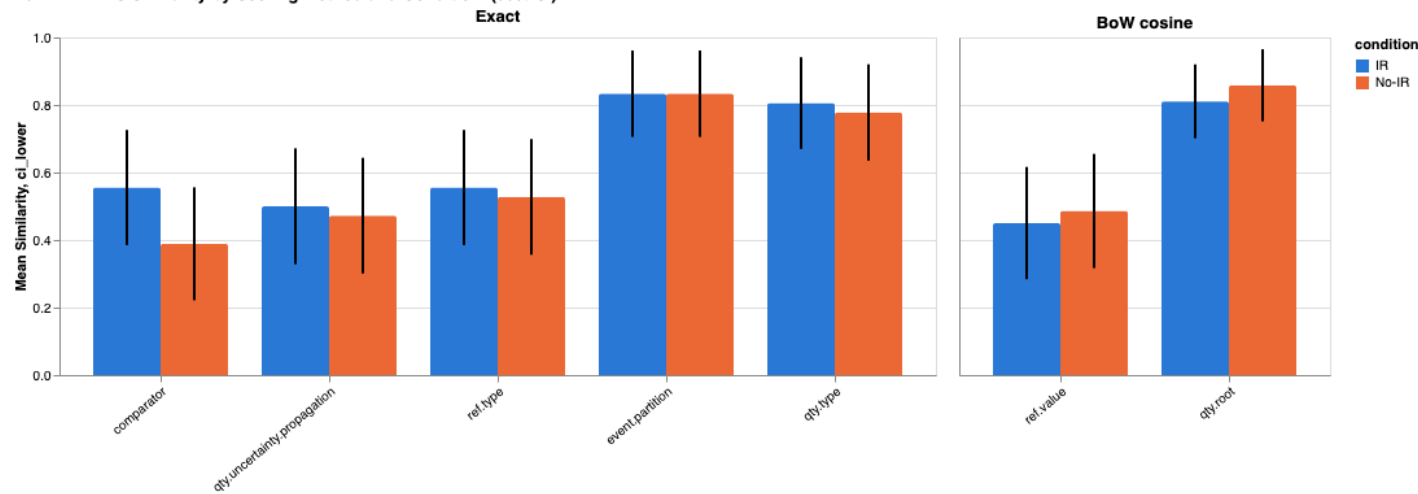
But the quantity itself is usually better preserved by the LLM alone with no grammar.

Some aspects of the quantity are better preserved by the grammar interventions.

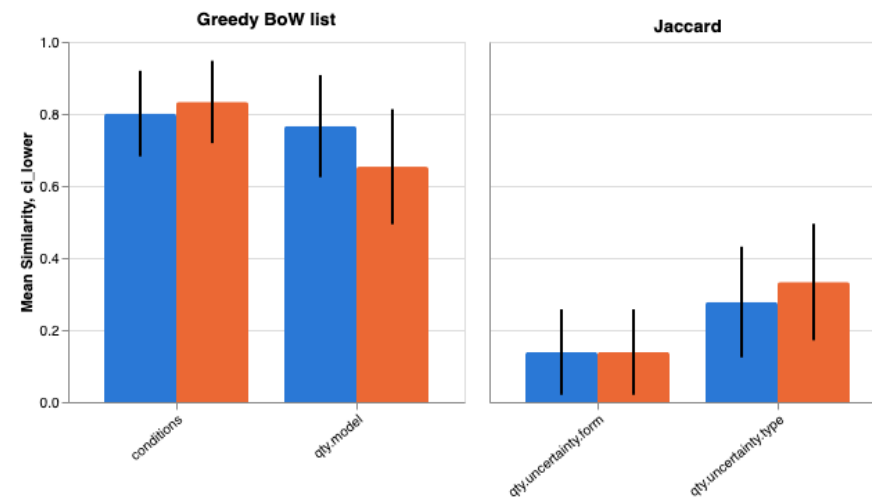


We can use semantic invariants to measure drift across many trials.

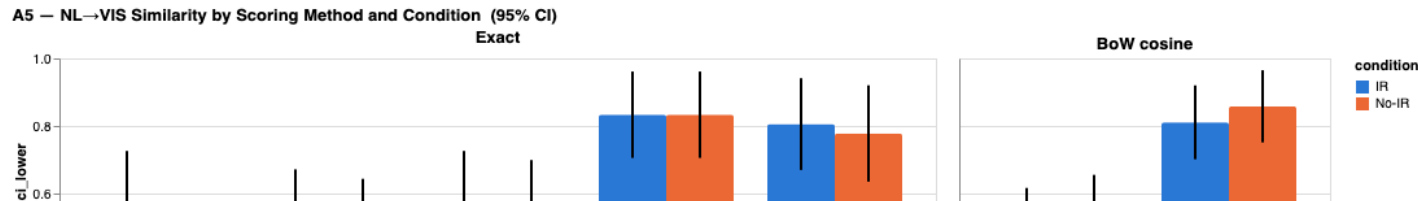
A5 — NL→VIS Similarity by Scoring Method and Condition (95% CI)



Referents and comparators are often better preserved by the grammar.

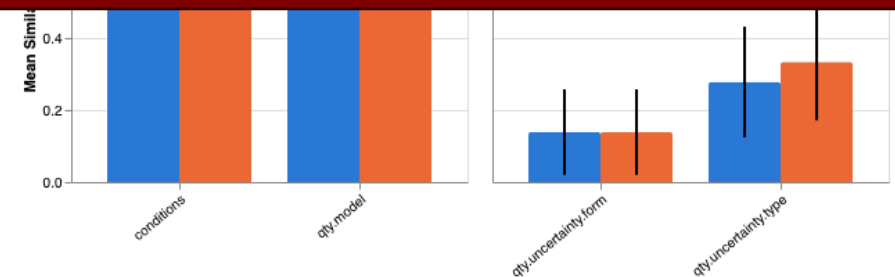


We can use semantic invariants to measure drift across many trials.



Grammar-based interventions to scaffold LLM thought show promise, but more work is needed.

Referents and comparators are often better preserved by the grammar.



Returning to the problems with naïve analysis approaches.

Manual data uploads to proprietary software platform.

I am going to give you a dataset. Can you please produce an altair visualization that allows me to test the following hypothesis 'Within comparable job sizes (grouped by quartiles of NODES_REQUESTED), median QUEUED_WAIT_SECONDS differs across QUEUE_NAME.' against this dataset.

Please share the dataset and I'll build the Altair visualization for you!

ANL-ALCF-DJC-POLARIS_20250101_20251231.csv

GZ

Ran 6 commands, viewed a file >

Now I have everything I need. Let me build a rich Altair visualization as an HTML artifact:

Created a file, read a file >

The visualization has two complementary panels:

Chart 1 — Grouped bar chart with IQR whiskers (log scale): Each queue is a group, bars are colored by node-size quartile, and the vertical rules show the interquartile range of wait times. This directly exposes whether median wait shifts within the same quartile as you move across queues.

Chart 2 — Heatmap (log color scale): Rows = queues, columns = node quartiles, color intensity = median wait. This makes cross-queue comparisons within each column (same job size) immediately visible.

Key findings supporting the hypothesis:

- Within Q4 (>2 nodes), `medium` has a median wait of ~20,861 s, while `debug-scaling`, `backfill-small`, and `demand` sit at 7-33 s — a ~3,000× difference.
- Within Q1 (<1 node), `preemptable` waits ~3,489 s vs. `debug / debug-scaling` at just 6 s.
- The hypothesis is strongly supported: queue identity matters far more than job size in determining wait time on Polaris.

Polaris wait viz

Code - HTML

Download

Download and manual integration into notebook and analysis workflows necessary.

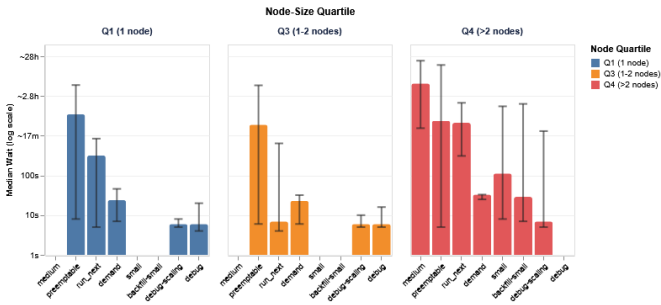
Polaris HPC — Queue Wait Time Analysis (2025)

ALCF Polaris - 262,700 jobs - Jan-Dec 2025 - Top 8 queues by volume

Hypothesis: Within comparable job sizes (grouped by quartiles of NODES_REQUESTED), median QUEUED_WAIT_SECONDS differs across

Chart 1 — Median Wait by Queue, Faceted by Node-Size Quartile (log scale)

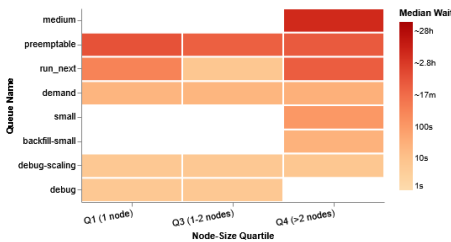
Each panel = one node-size quartile. Compare bar heights across queues within the same panel to test the hypothesis. Error bars = IQR (Q25-Q75). How



Y-axis shows $\log_{10}$ (median wait in seconds). Tick labels are back-transformed to seconds for readability.
Node quartiles: Q1 = 1 node · Q3 = 1-2 nodes · Q4 = >2 nodes (Q1=Q2 due to skew — 75% of all jobs request just 1 node).
`medium` queue only appears in Q4 (all its jobs request >2 nodes).

Chart 2 — Heatmap: Median Wait across Queue × Node Quartile

Color = $\log$ (median wait). Dark red = long wait. Empty cells = queue has no jobs in that size bucket.



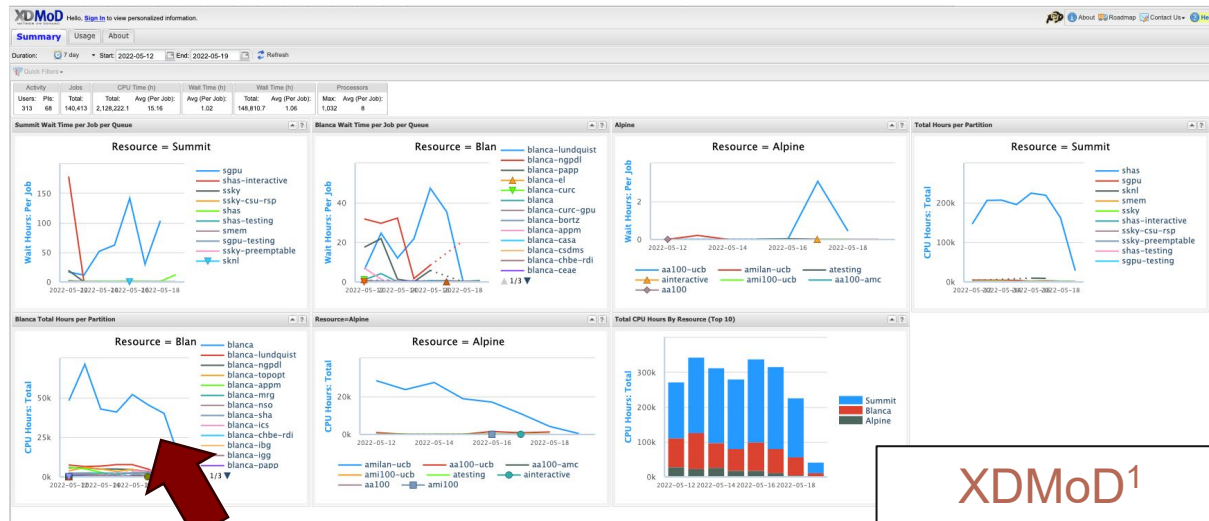
No guarantees of translation fidelity between natural language and visual representation.

No provenance on production of visualizations and report.



Part 2: Integrating Effective Visualization into Common Workflows

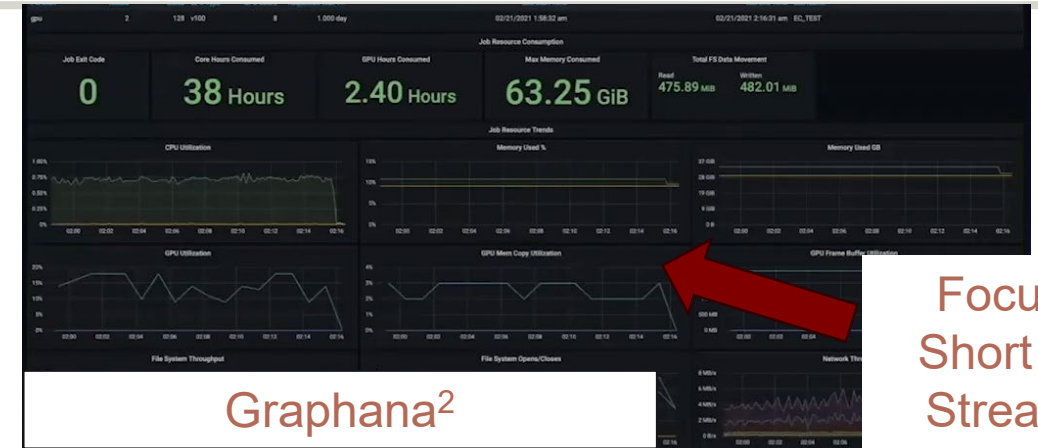
HPC Systems Analysis and Research Currently



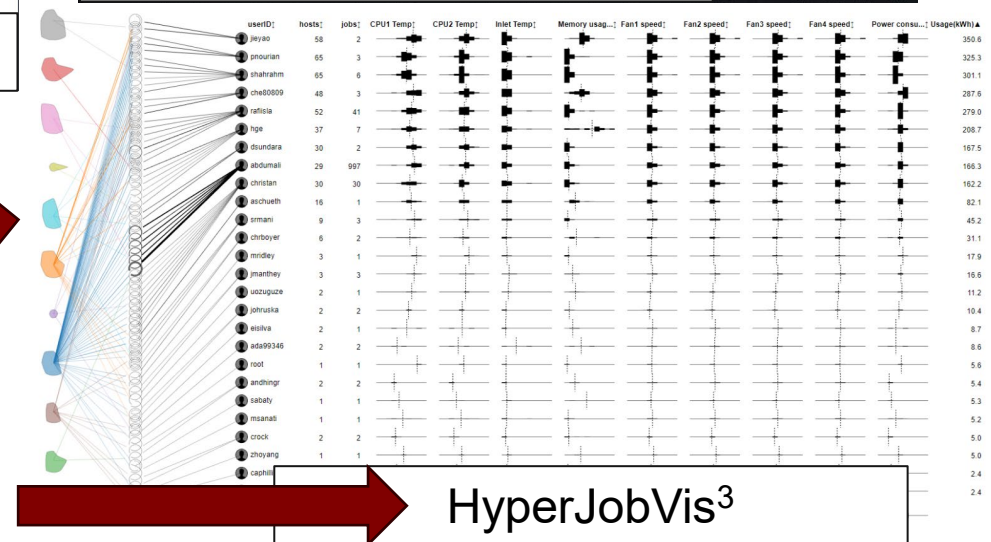
Individual Line Charts Show Trends but Elide Distributions

Using basic visual idioms that can be supported by any out of the box VIS software in a Notebook

Conditionally-useful fixed organization of jobs
Not designed or evaluated using formal HCC methods



Focus on Short Term Streaming Data



1. Furlani, Thomas & Schneider, Barry & Jones, Matthew & Towns, John & Hart, David & Gallo, Steven & Deleon, Robert & Lu, Charng-Da & Ghadersohi, Amin & Gentner, Ryan & Patra, Abani & von Laszewski, Gregor & Wang, Fugang & Palmer, Jeffrey & Simakov, Nikolay. (2013). Using XDMoD to facilitate XSEDE operations, planning and analysis. ACM International Conference Proceeding Series. 10.1145/2484762.2484763.
2. Labs, Grafana. "Grafana Documentation." 2018. <https://grafana.com/docs/>.
3. Nguyen, Ngan, Tommy Dang, Jon Hass, and Yong Chen. "Hyperjobviz: Visualizing resource allocations in high-performance computing center via multivariate health-status data." In 2019 IEEE/ACM Industry/University Joint International Workshop on Data-center Automation, Analytics, and Control (DAAC), pp. 19-24. IEEE, 2019.

HPC Systems Analysis and Research Currently

Scripting Space



```

[1]: %env ANYWIDGET_HMR=1
import pandas as pd
from guidepost import Guidepost
from IPython.display import HTML
import json

# jobs_data = pd.read_csv("../data/ANL-ALCF-DJC-POLARIS_20250101_20251231.csv.gz", compression='gzip')

jobs_data = pd.read_parquet("../data/esif.hpc.kestrel.job-anon/year=2025/month=1/kestrel_jobs_202501_0.parquet")
# jobs_data = pd.concat([jobs_data, pd.read_parquet("../data/esif.hpc.kestrel.job-anon/year=2025/month=2/kestrel_jobs_202502_0.parquet")])
# Randomly retain 50% of the combined rows
# jobs_data = jobs_data.sample(
#     # frac=0.5,
#     # random_state=42,
#     # ).reset_index(drop=True)

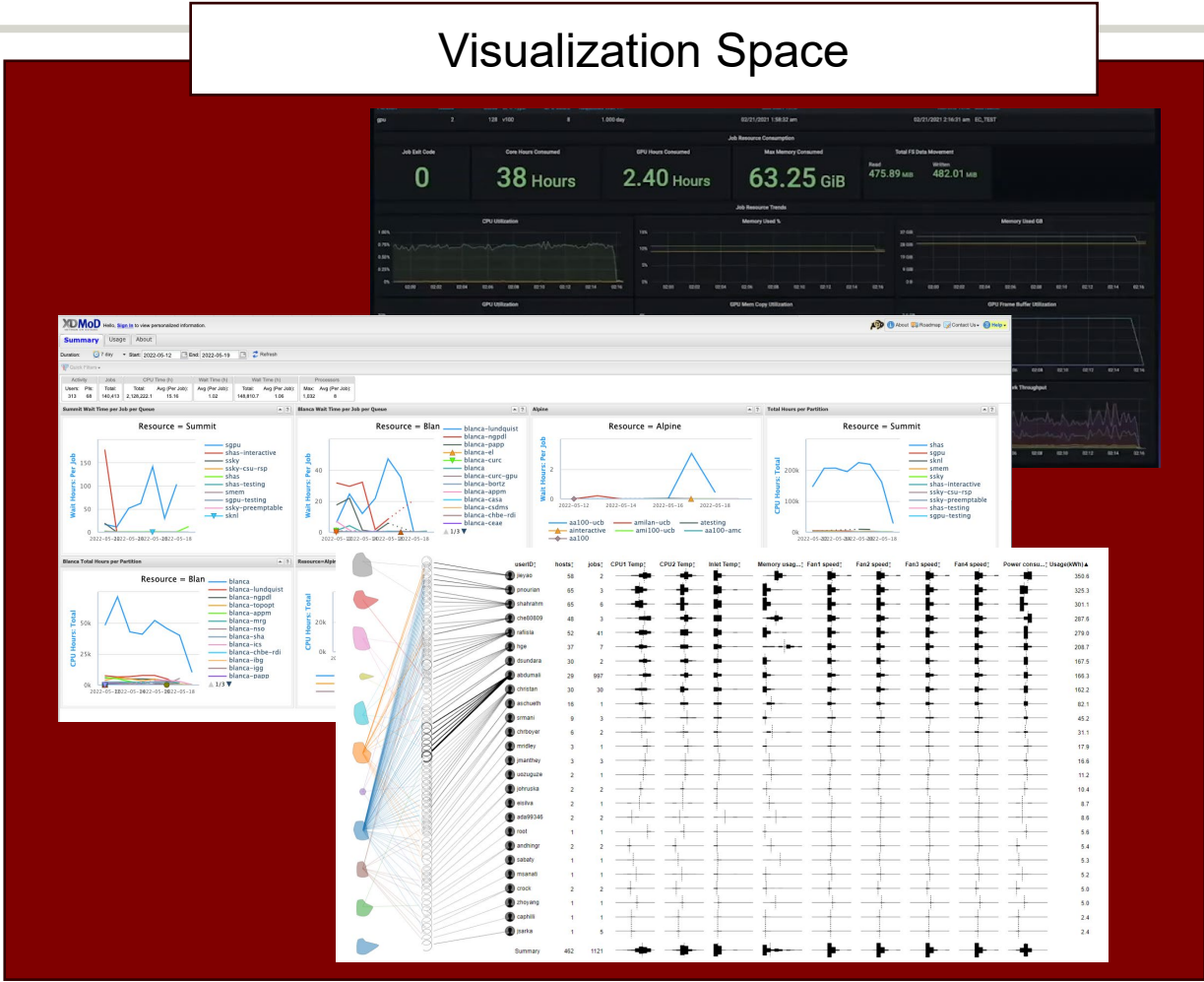
env: ANYWIDGET_HMR=1

[2]: jobs_data.head(5).T

[2]:
```

	0	1	2	3	4
id	6488862	6488861	6488860	6488869	6488844
job_id	6488862	6488861	6488860	6488869	6488844
array_pos	NaN	NaN	NaN	NaN	NaN
array_range	NaN	NaN	NaN	NaN	NaN
name_hash	0e72d17	0e72d17	0e72d17	0e72d17	0e72d17
user_hash	ce1d591	ce1d591	ce1d591	ce1d591	ce1d591
account_hash	341be45	341be45	341be45	341be45	341be45
submit_line_hash	5e9d770	5e9d770	5e9d770	5e9d770	5e9d770

Visualization Space



HPC Systems Analysis and Research Currently

Scripting Space

Visualization Space

Most
Complicated
Analysis
Requires Both



```
Launcher x demo_notebook.ipynb x
Code
Notebook Python 3 (ipykernel)

[1]: %env ANYWIDGET_HMR=1
import pandas as pd
from guido import Guidopost
from IPython.display import HTML
import json

# jobs_data = pd.read_csv("../data/ANL-ALCF-DJC-POLARIS_20250101_20251231.csv.gz", compression='gzip')

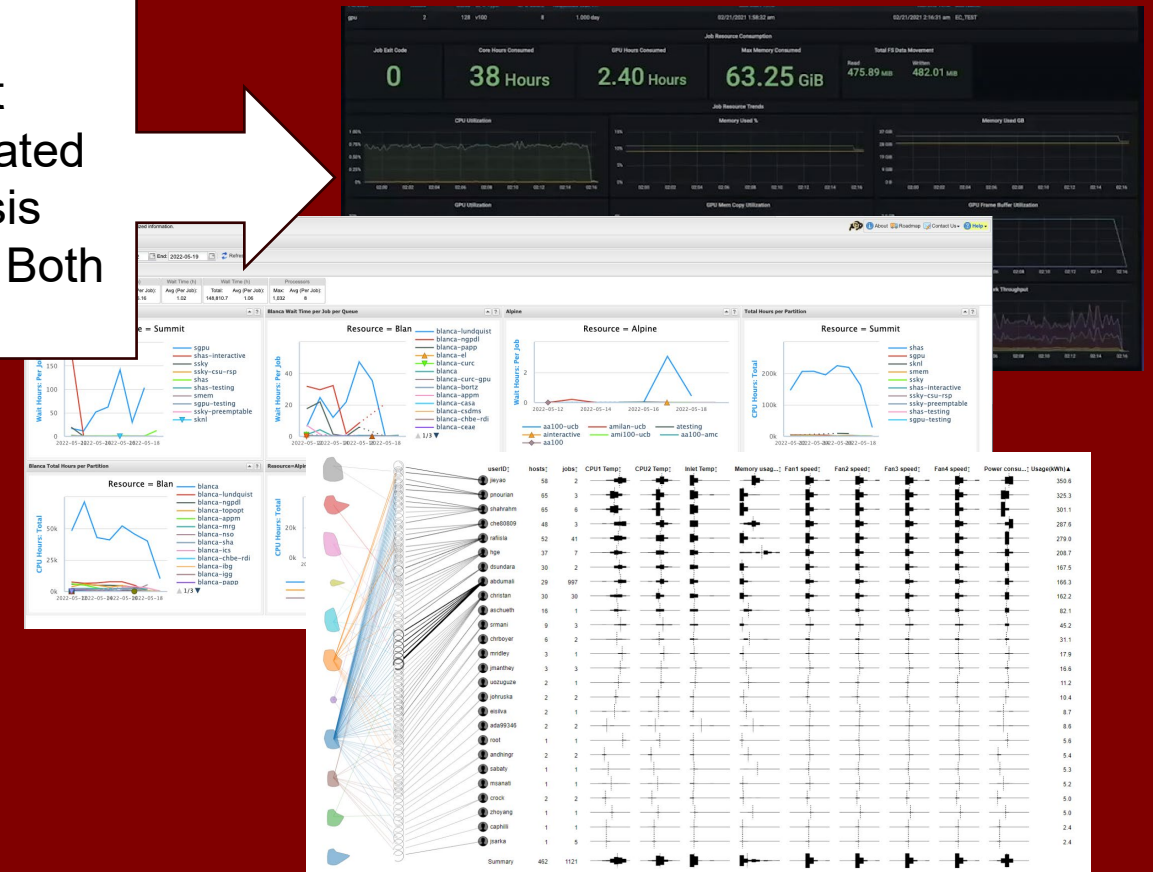
jobs_data = pd.read_parquet("../data/esif.hpc.kestrel.job-anon/year=2025/month=1/kestrel_jobs_202501_0.parquet")
# jobs_data = pd.concat([jobs_data, pd.read_parquet("../data/esif.hpc.kestrel.job-anon/year=2025/month=2/kestrel_jobs_202502_0.parquet")])
# Randomly retain 50% of the combined rows
# jobs_data = jobs_data.sample(
#     # frac=0.3,
#     # random_state=42,
#     # ).reset_index(drop=True)

env: ANYWIDGET_HMR=1

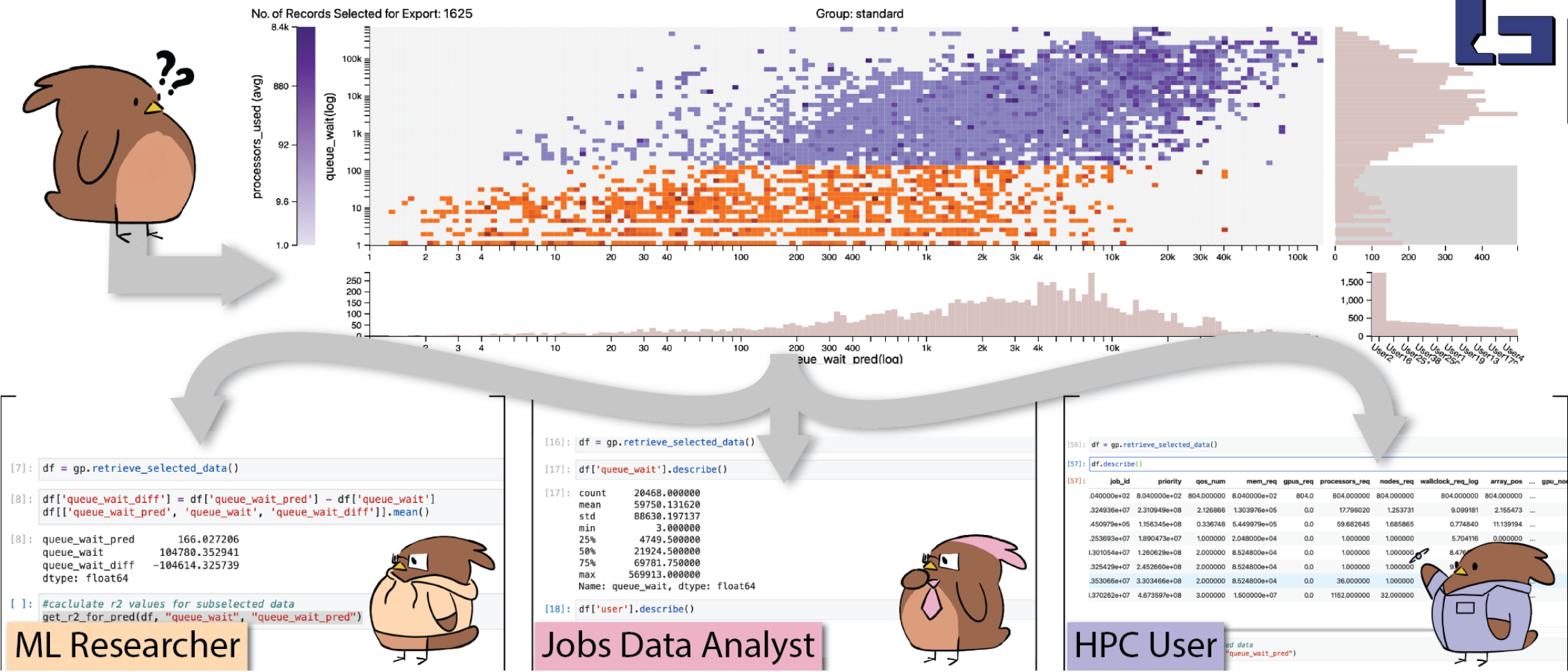
[2]: jobs_data.head(5).T

[2]:
```

	0	1	2	3	4
id	6488862	6488861	6488860	6488869	6488844
job_id	6488862	6488861	6488860	6488869	6488844
array_pos	NaN	NaN	NaN	NaN	NaN
array_range	NaN	NaN	NaN	NaN	NaN
name_hash	0e72d17	0e72d17	0e72d17	0e72d17	0e72d17
user_hash	ce1d591	ce1d591	ce1d591	ce1d591	ce1d591
account_hash	341be45	341be45	341be45	341be45	341be45
submit_line_hash	5e9d770	5e9d770	5e9d770	5e9d770	5e9d770

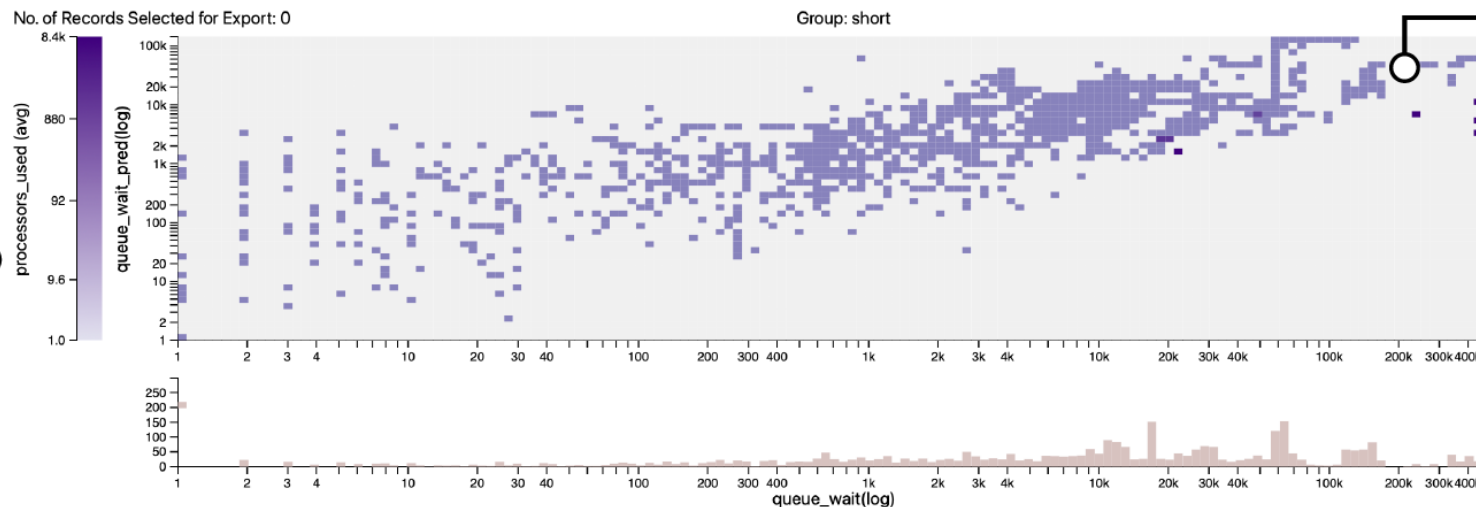


Spanning Visualization and Scripting with Guidepost

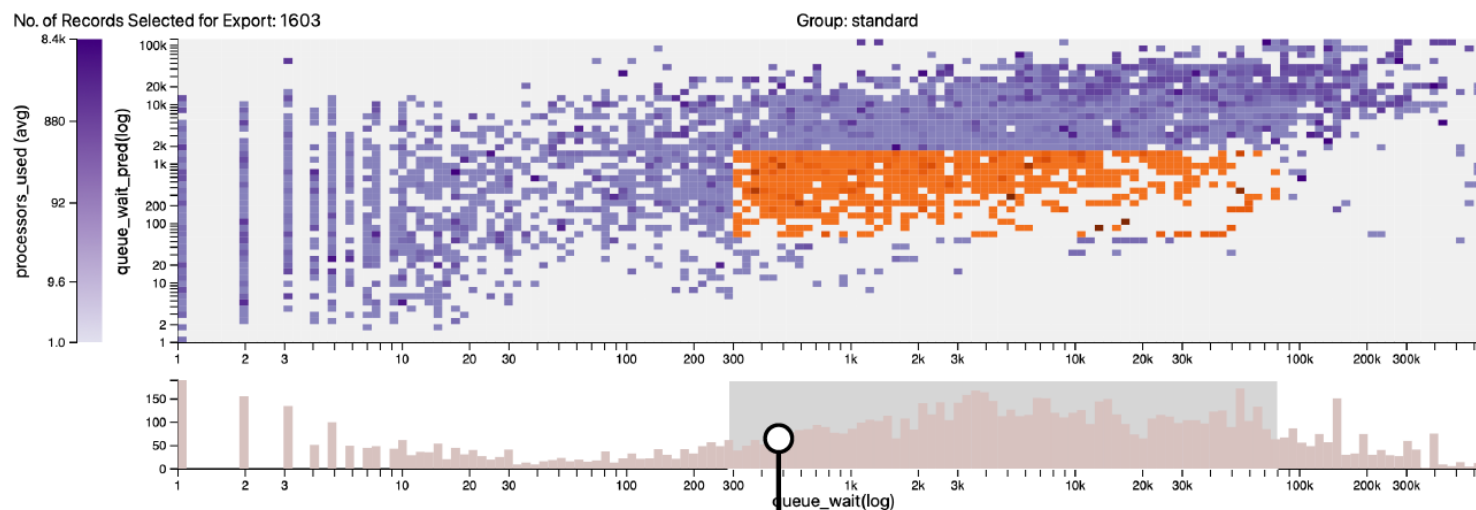
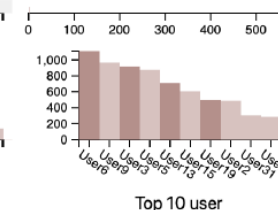


Guidepost Overview

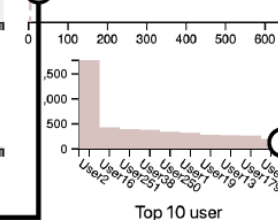
Small
Multiples



Main Summary
View



Categorical
Data View



Selectable Histograms



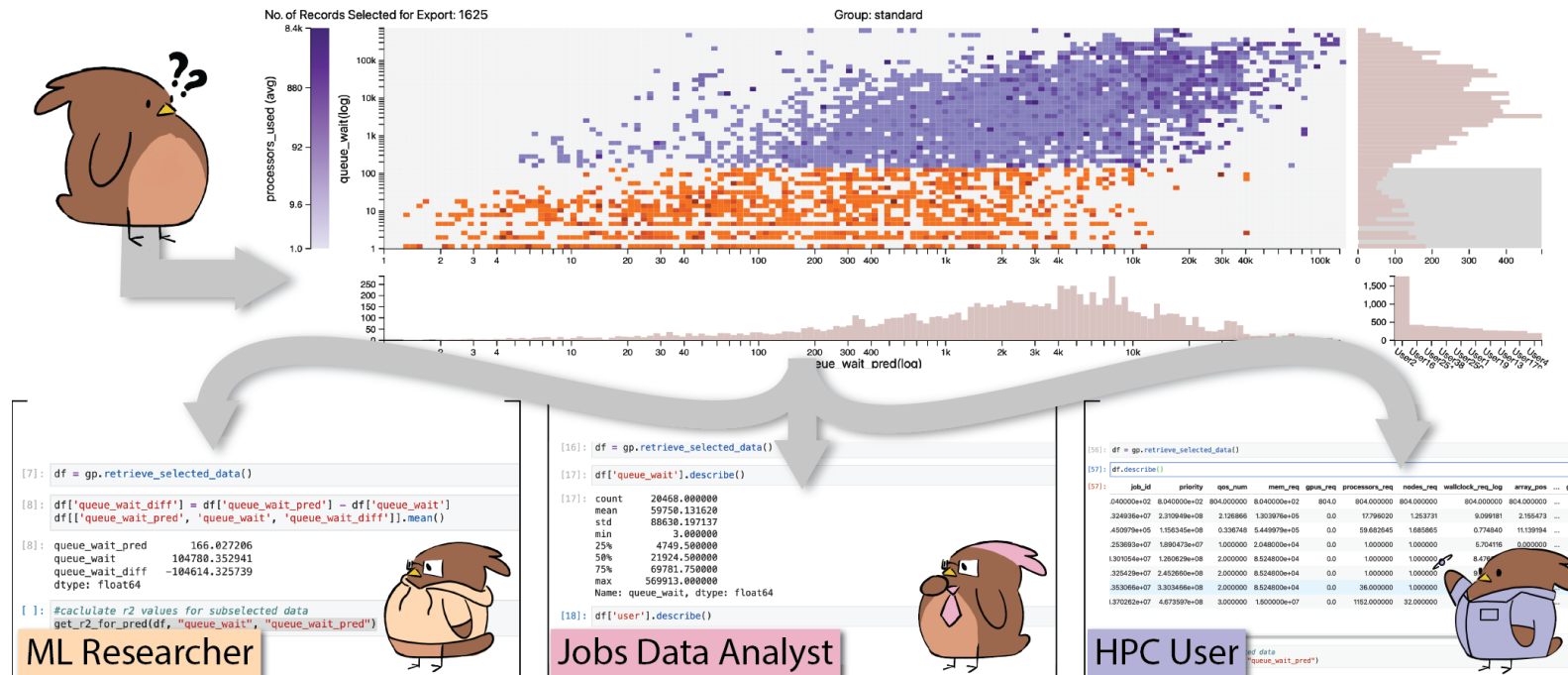
Guidepost – Selection

C



Guidepost – Export

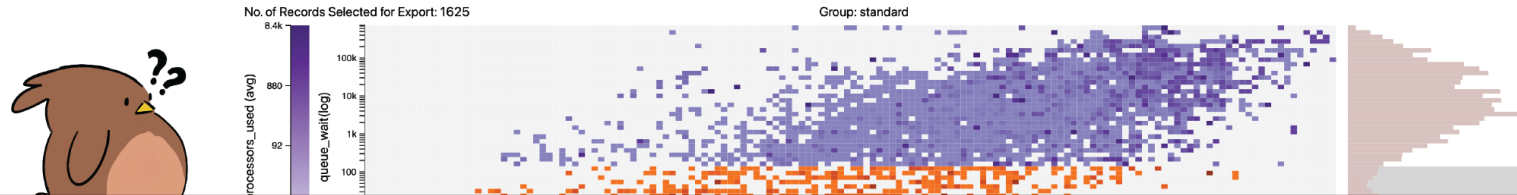
Where Guidepost Fails, and AI Naturally Fits In.



Guidepost only allows for the interactive retrieval of “interesting” data.

Each user group’s analysis still needs to be built up manually.

Where Guidepost Fails, and AI Naturally Fits In.



When a user identifies an interesting phenomenon in the data, we should provide tools that help answer focused questions about it.

Guidepost only allows for the interactive retrieval of “interesting” data.

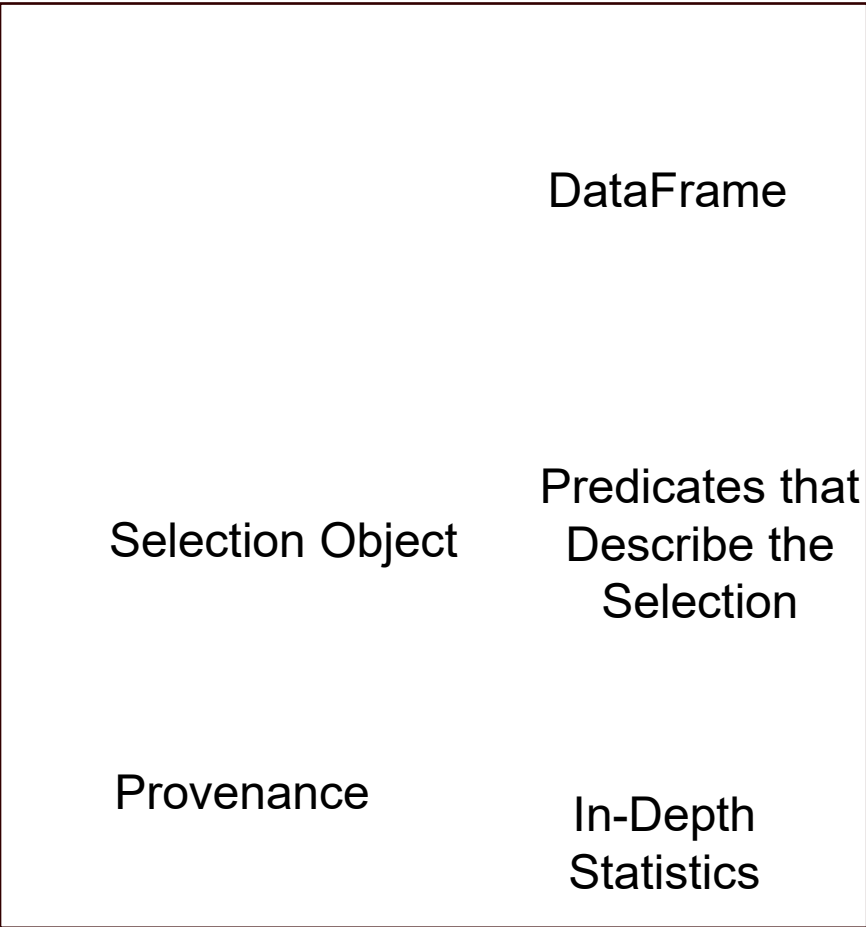
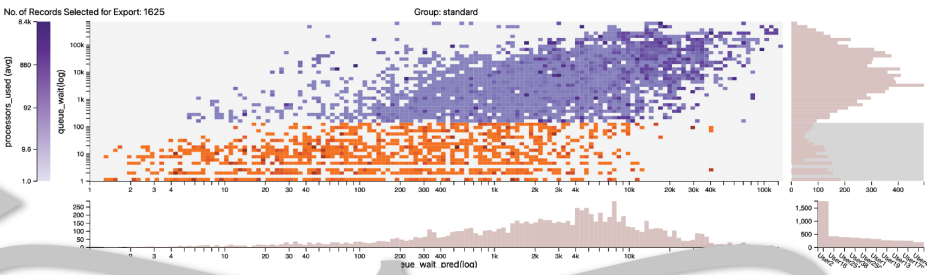
Each user group’s analysis still needs to be built up manually.

Designing AI integration for notebook-based workflows.

Input Data



Interactive Visualization with Export



LLM w/ Focused Question

DataFrame

Selection Object

Predicates that Describe the Selection

Provenance

In-Depth Statistics

Output New Analysis Cells in Place

Designing AI integration for notebook-based workflows.

DataFrame

Selection
Object

Predicates that
Describe the
Selection

Provenance

In-Depth
Statistics

Designing AI integration for notebook-based workflows.

DataFrame

Predicates that Describe the Selection

Selection
Object

Provenance

In-Depth
Statistics

```
[8]: # A human-readable characterization: size, the predicate, and headline shifts.  
print(sel.summary())
```

Selection: 21,896 of 400,896 records (5.5% of population)

Predicate:

- partition ∈ {standard-stdby, standard, gpu-h100-stdby, shared, ...(+14)}
- state ∈ {COMPLETED}
- state_simple ∈ {COMPLETED}
- submit_time ∈ [2025-01-26T00:05:40+00:00, 2025-01-29T23:59:26+00:00]
- start_time ∈ [2025-01-26T00:19:47+00:00, 2025-03-03T02:11:39+00:00]
- end_time ∈ [2025-01-26T00:35:55+00:00, 2025-03-03T19:30:12+00:00]
- processors_used ∈ [1, 6,656]
- day ∈ [25, 29]

Notable:

- processors_used: median 3.25× population (0.03σ shift)
- day: median 1.42× population (1.107σ shift)

```
[27]: # Operationalize the *same* predicate three ways – hand it to pandas, SQL, or Altair:  
print("pandas :", sel.to_pandas(), "\n")  
print("sql      :", sel.to_sql_where(), "\n")  
flt = sel.to_altair_filter()
```

```
pandas : (`partition` in ['standard-stdby', 'standard', 'gpu-h100-stdby', 'shared', 'sh  
'long-stdby', 'bigmem', 'debug-gpu', 'short-stdby', 'gpu-h100l', 'project_3', 'debug-st  
in ['COMPLETED', 'FAILED', 'CANCELLED', 'TIMEOUT', 'OUT_OF_MEMORY', 'NODE_FAIL']) and (  
_time` <= @pd.Timestamp("2025-01-29T23:59:26+00:00")) and (`end_time` >= @pd.Timestamp(  
3T19:30:12+00:00")) and (`day` >= 6.0 and `day` <= 29.0)
```

Designing AI integration for notebook-based workflows.

DataFrame

```
[8]: # A human-readable characterization: size, the predicate, and headline shifts.
print(sel.summary())

Selection: 21,896 of 400,896 records (5.5% of population)
Predicate:
  • partition ∈ {standard-stdby, standard, gpu-h100-stdby, shared, ...(+14)}
  • state ∈ {COMPLETED}
  • state_simple ∈ {COMPLETED}
  • submit_time ∈ [2025-01-26T00:05:40+00:00, 2025-01-29T23:59:26+00:00]
  • start_time ∈ [2025-01-26T00:19:47+00:00, 2025-03-03T02:11:39+00:00]
  • end_time ∈ [2025-01-26T00:35:55+00:00, 2025-03-03T19:30:12+00:00]
  • processors_used ∈ [1, 6,656]
  • day ∈ [25, 29]
Notable:
  • processors_used: median 3.25× population (0.03σ shift)
  • day: median 1.42× population (1.107σ shift)
```

Selection
Object

Predicates that Describe the
Selection

Provenance

In-Depth Statistics

```
[10]: # Rich distribution of one variable, selection vs. population
# (quantiles, skew/kurtosis, a normality read, a sparkline, and the shift vs. the whole).
print(sel.describe_distribution("cpu_eff"))
```

Distribution of 'cpu_eff' (selection vs. population)

n	30145	(7.5% of population)
mean	0.969	(pop 0.385)
median	0.99	(pop 0.039)
std	0.141	(pop 0.447)
min / max	1e-06 / 0.997	
q25 / q75	0.989 / 0.991	(IQR 0.002)
skew	-6.66	kurtosis 42.461
normality	likely non-normal (normaltest p=0)	
vs pop	median 25.16×, shift 1.306σ	
histogram		

```
[11]: sel.describe_distribution("state_simple")
```

Categorical distribution of 'state_simple' (selection vs. population)

n	30145	(7.5% of population)	
distinct	3	(pop 7)	
concentration	0.221	(0 = one value, 1 = spread evenly across its values)	
top categories	selection	population	lift
COMPLETED	93.5%	50.4%	1.9×
TIMEOUT	6.5%	4.4%	1.5×
FAILED	0.0%	23.5%	0.0×
over-represented:	COMPLETED (1.9×)		

Designing AI integration for notebook-based workflows.

DataFrame

```
[8]: # A human-readable characterization: size, the predicate, and headline shifts.
print(sel.summary())

Selection: 21,896 of 400,896 records (5.5% of population)
Predicate:
  • partition ∈ {standard-stdby, standard, gpu-h100-stdby, shared, ...(+14)}
  • state ∈ {COMPLETED}
  • state_simple ∈ {COMPLETED}
  • submit_time ∈ [2025-01-26T00:05:40+00:00, 2025-01-29T23:59:26+00:00]
  • start_time ∈ [2025-01-26T00:19:47+00:00, 2025-03-03T02:11:39+00:00]
  • end_time ∈ [2025-01-26T00:35:55+00:00, 2025-03-03T19:30:12+00:00]
  • processors_used ∈ [1, 6,656]
  • day ∈ [25, 29]
Notable:
  • processors_used: median 3.25× population (0.03o shift)
  • days: median 1.42× population (1.107o shift)
```

Selection
Object

Predicates that Describe the Selection

```
[10]: # Rich distribution of one variable, selection vs. population
# quantiles, skew/kurtosis, a normality read, a sparkline, and the shift vs. the whole).
print(sel.describe_distribution("cpu_eff"))

Distribution of 'cpu_eff' (selection vs. population)
n 30145 (7.5% of population)
mean 0.969 (pop 0.385)
median 0.99 (pop 0.039)
std 0.141 (pop 0.447)
min / max 1e-06 / 0.997
q25 / q75 0.989 / 0.991 (IQR 0.002)
skew -6.06 kurtosis 42.461
normality likely non-normal (normaltest p=0)
vs pop median 25.16×, shift 1.306o
histogram
```

```
[11]: sel.describe_distribution("state_simple")

[11]: Categorical distribution of 'state_simple' (selection vs. population)
n 30145 (7.5% of population)
distinct 3 (pop 7)
concentration 0.221 (0 = one value, 1 = spread evenly across its values)
top categories selection population lift
COMPLETED 91.5% 58.4% 1.9x
TIMEOUT 6.5% 4.4% 1.5x
FAILED 0.0% 23.5% 0.0x
over-represented: COMPLETED (1.9x)
```

Provenance

In-Depth Statistics

Designing AI integration for notebook-based workflows.

DataFrame

```
[8]: # A human-readable characterization: size, the predicate, and headline shifts.
print(sel.summary())

Selection: 21,896 of 400,896 records (5.5% of population)
Predicate:
  • partition ∈ {standard-stdby, standard, gpu-h100-stdby, shared, ...(+14)}
  • state ∈ {COMPLETED}
  • state_simple ∈ {COMPLETED}
  • submit_time ∈ [2025-01-26T00:05:40+00:00, 2025-01-29T23:59:26+00:00]
  • start_time ∈ [2025-01-26T00:19:47+00:00, 2025-03-03T02:11:39+00:00]
  • end_time ∈ [2025-01-26T00:35:55+00:00, 2025-03-03T19:30:12+00:00]
  • processors_used ∈ [1, 6,656]
  • day ∈ [25, 29]
Notable:
  • processors_used: median 3.25× population (0.03o shift)
  • days: median 1.42× population (1.107o shift)
```

Selection
Object

Predicates that Describe the Selection

```
[10]: # Rich distribution of one variable, selection vs. population
# quantiles, skew/kurtosis, a normality read, a sparkline, and the shift vs. the whole.
print(sel.describe_distribution("cpu_eff"))

Distribution of 'cpu_eff' (selection vs. population)
n          30145 (7.5% of population)
mean       0.969 (pop 0.385)
median     0.99 (pop 0.839)
std        0.141 (pop 0.447)
min / max  1e-06 / 0.997
q25 / q75  0.989 / 0.991 (IQR 0.002)
skew       -6.06 kurtosis 42.461
normality  likely non-normal (normaltest p=0)
vs pop     median 25.16×, shift 1.306o
histogram  

[11]: sel.describe_distribution("state_simple")

[11]: Categorical distribution of 'state_simple' (selection vs. population)
n          30145 (7.5% of population)
distinct    3 (pop 7)
concentration 0.221 (0 = one value, 1 = spread evenly across its values)
top categories selection population lift
COMPLETED    91.5%    58.4%    1.9x
TIMEOUT        6.5%     4.4%    1.5x
FAILED         0.0%    23.5%    0.0x
over-represented: COMPLETED (1.9x)
```

Provenance

In-Depth Statistics

“Do busy weeks on the system have higher failure rates for jobs?”

LLM Calling Interface

Designing AI integration for notebook-based workflows.

DataFrame

```
[8]: # A human-readable characterization: size, the predicate, and headline shifts.
print(sel.summary())

Selection: 21,896 of 400,896 records (5.5% of population)
Predicate:
  • partition ∈ {standard-stdby, standard, gpu-h100-stdby, shared, ...(+14)}
  • state ∈ {COMPLETED}
  • state_simple ∈ {COMPLETED}
  • submit_time ∈ [2025-01-26T00:05:40+00:00, 2025-01-29T23:59:26+00:00]
  • start_time ∈ [2025-01-26T00:19:47+00:00, 2025-03-03T02:11:39+00:00]
  • end_time ∈ [2025-01-26T00:35:55+00:00, 2025-03-03T19:30:12+00:00]
  • processors_used ∈ [1, 6,656]
  • day ∈ [25, 29]
Notable:
  • processors_used: median 3.25× population (0.03o shift)
  • days: median 1.42× population (1.107o shift)
```

Selection
Object

Predicates that Describe the
Selection

```
[10]: # Rich distribution of one variable, selection vs. population
# quantiles, skew/kurtosis, a normality read, a sparkline, and the shift vs. the whole.
print(sel.describe_distribution("cpu_eff"))

Distribution of 'cpu_eff' (selection vs. population)
n 30145 (7.5% of population)
mean 0.969 (pop 0.385)
median 0.99 (pop 0.839)
std 0.141 (pop 0.447)
min / max 1e-06 / 0.997
q25 / q75 0.989 / 0.991 (IQR 0.002)
skew -6.06 kurtosis 42.461
normality likely non-normal (normaltest p=0)
vs pop median 25.16×, shift 1.306o
histogram
```

```
[11]: sel.describe_distribution("state_simple")

[11]: Categorical distribution of 'state_simple' (selection vs. population)
n 30145 (7.5% of population)
distinct 3 (pop 7)
concentration 0.221 (0 = one value, 1 = spread evenly across its values)
top categories selection population lift
COMPLETED 91.5% 58.4% 1.9x
TIMEOUT 6.5% 4.4% 1.5x
FAILED 0.0% 23.5% 0.0x
over-represented: COMPLETED (1.9x)
```

Provenance

In-Depth Statistics

“Do busy weeks on the system have higher failure rates for jobs?”

selection.analyze()

Designing AI integration for notebook-based workflows.

“Do busy weeks on the system have higher failure rates for jobs?”

DataFrame

```
[8]: # A human-readable characterization: size, the predicate, and headline shifts.
print(sel.summary())

Selection: 21,896 of 400,896 records (5.5% of population)
Predicate:
  • partition ∈ {standard-stdby, standard, gpu-h100-stdby, shared, ...(+14)}
  • state ∈ {COMPLETED}
  • state_simple ∈ {COMPLETED}
  • submit_time ∈ [2025-01-26T00:05:40+00:00, 2025-01-29T23:59:26+00:00]
  • start_time ∈ [2025-01-26T00:19:47+00:00, 2025-03-03T02:11:39+00:00]
  • end_time ∈ [2025-01-26T00:35:55+00:00, 2025-03-03T19:30:12+00:00]
  • processors_used ∈ [1, 6,656]
  • day ∈ [25, 29]
Notable:
  • processors_used: median 3.25× population (0.03o shift)
  • days: median 1.42× population (1.107o shift)
```

Selection
Object

Predicates that Describe the Selection

```
[10]: # Rich distribution of one variable, selection vs. population
# quantiles, skew/kurtosis, a normality read, a sparkline, and the shift vs. the whole).
print(sel.describe_distribution("cpu_eff"))

Distribution of 'cpu_eff' (selection vs. population)
n 30145 (7.5% of population)
mean 0.969 (pop 0.385)
median 0.99 (pop 0.839)
std 0.141 (pop 0.447)
min / max 1e-06 / 0.997
q25 / q75 0.989 / 0.991 (IQR 0.002)
skew -6.06 kurtosis 42.461
normality likely non-normal (normaltest p=0)
vs pop median 25.16×, shift 1.306o
histogram
```

```
[11]: sel.describe_distribution("state_simple")

[11]: Categorical distribution of 'state_simple' (selection vs. population)
n 30145 (7.5% of population)
distinct 3 (pop 7)
concentration 0.221 (0 = one value, 1 = spread evenly across its values)
top categories selection population lift
COMPLETED 91.5% 58.4% 1.9x
TIMEOUT 6.5% 4.4% 1.5x
FAILED 0.0% 23.5% 0.0x
over-represented: COMPLETED (1.9x)
```

Provenance

In-Depth Statistics

selection.analyze()

VIS Code Cell

Designing AI integration for notebook-based workflows.

“Do busy weeks on the system have higher failure rates for jobs?”

DataFrame

```
[8]: # A human-readable characterization: size, the predicate, and headline shifts.
print(sel.summary())

Selection: 21,896 of 400,896 records (5.5% of population)
Predicate:
  • partition ∈ {standard-stdby, standard, gpu-h100-stdby, shared, ...(+14)}
  • state ∈ {COMPLETED}
  • state_simple ∈ {COMPLETED}
  • submit_time ∈ [2025-01-26T00:05:40+00:00, 2025-01-29T23:59:26+00:00]
  • start_time ∈ [2025-01-26T00:19:47+00:00, 2025-03-03T02:11:39+00:00]
  • end_time ∈ [2025-01-26T00:35:55+00:00, 2025-03-03T19:30:12+00:00]
  • processors_used ∈ [1, 6,656]
  • day ∈ [25, 29]
Notable:
  • processors_used: median 3.25× population (0.03o shift)
  • days: median 1.42× population (1.107o shift)
```

Selection
Object

Predicates that Describe the Selection

```
[10]: # Rich distribution of one variable, selection vs. population
# quantiles, skew/kurtosis, a normality read, a sparkline, and the shift vs. the whole).
print(sel.describe_distribution("cpu_eff"))

Distribution of 'cpu_eff' (selection vs. population)
n 30145 (7.5% of population)
mean 0.969 (pop 0.385)
median 0.99 (pop 0.839)
std 0.141 (pop 0.447)
min / max 1e-06 / 0.997
q25 / q75 0.989 / 0.991 (IQR 0.002)
skew -6.66 kurtosis 42.461
normality likely non-normal (normaltest p=0)
vs pop median 25.16×, shift 1.306o
histogram
```

```
[11]: sel.describe_distribution("state_simple")

[11]: Categorical distribution of 'state_simple' (selection vs. population)
n 30145 (7.5% of population)
distinct 3 (pop 7)
concentration 0.221 (0 = one value, 1 = spread evenly across its values)
top categories selection population lift
COMPLETED 91.5% 58.4% 1.9x
TIMEOUT 6.5% 4.4% 1.5x
FAILED 0.0% 23.5% 0.0x
over-represented: COMPLETED (1.9x)
```

Provenance

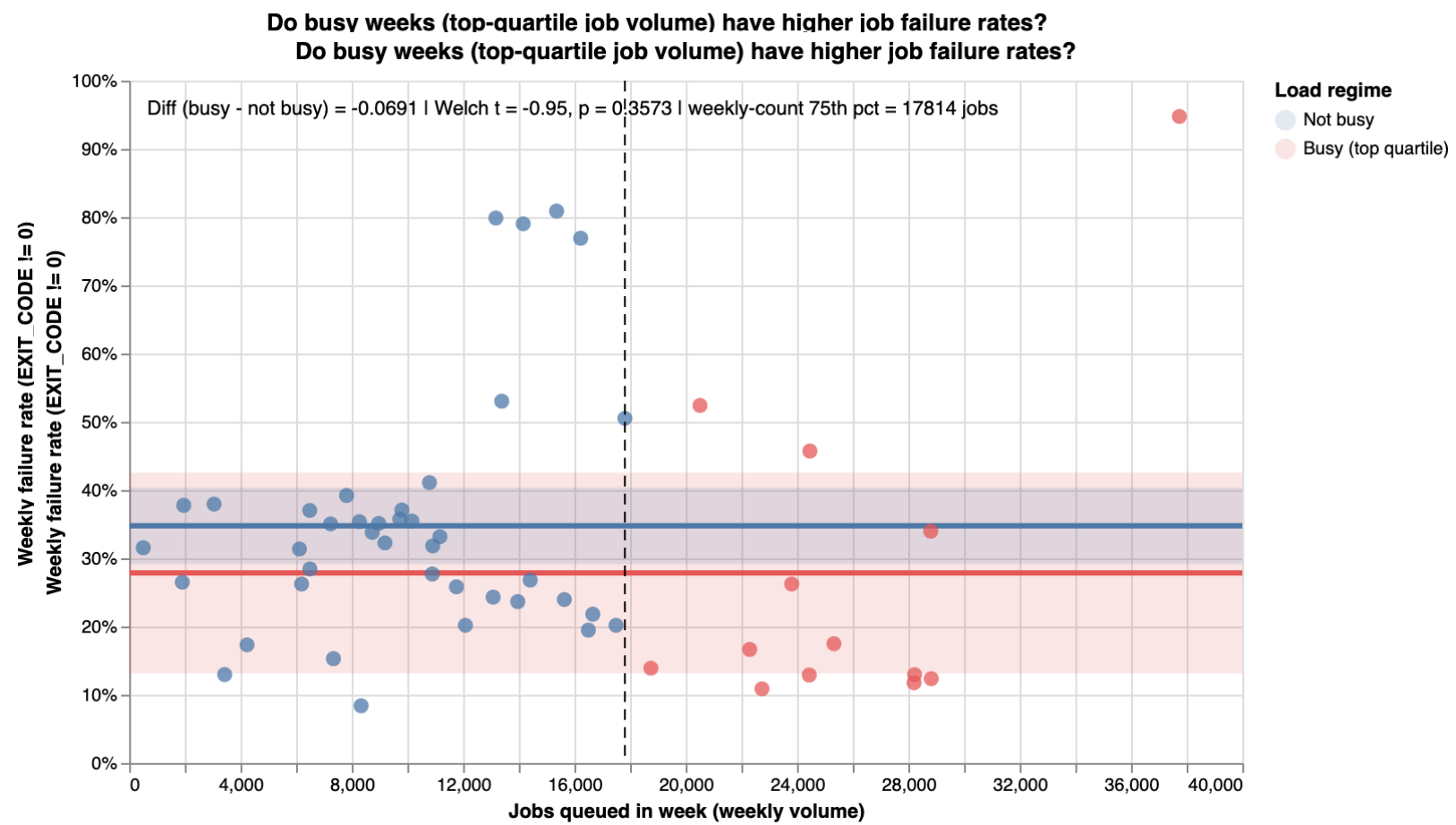
In-Depth Statistics



VIS Code Cell

Designing AI integration for notebook-based workflows.

“Do busy weeks on the system have higher failure rates for jobs?”





Conclusions

Our two barriers and the solutions.

Download and manual integration into notebook and analysis workflows. Manual data uploads to proprietary software platform. No provenance on production of visualizations and report.

No guarantees of accurate translation between natural language and visual representation.

Workflow Integration

- Provide tools for users to move from exploratory analysis to focused hypothesis testing.
- Design tools to work with analysis platforms like Jupyter



Translation Fidelity

- Scaffolding LLM thought with formal grammars shows promise but needs more work
- Semantic invariants allow us to measure systematically how translations drift across representations

Thank you! :D



Kate Isaacs



Andrew McNutt



Alex Kale



Olga Pearce



Stephanie Brink



Michela Taufer



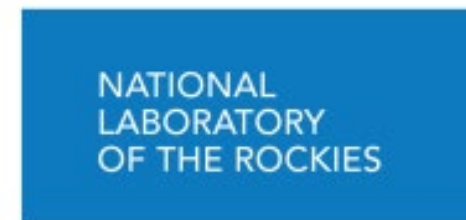
Dmitry Duplyakin



Kevin Menear



**Hewlett Packard
Enterprise**



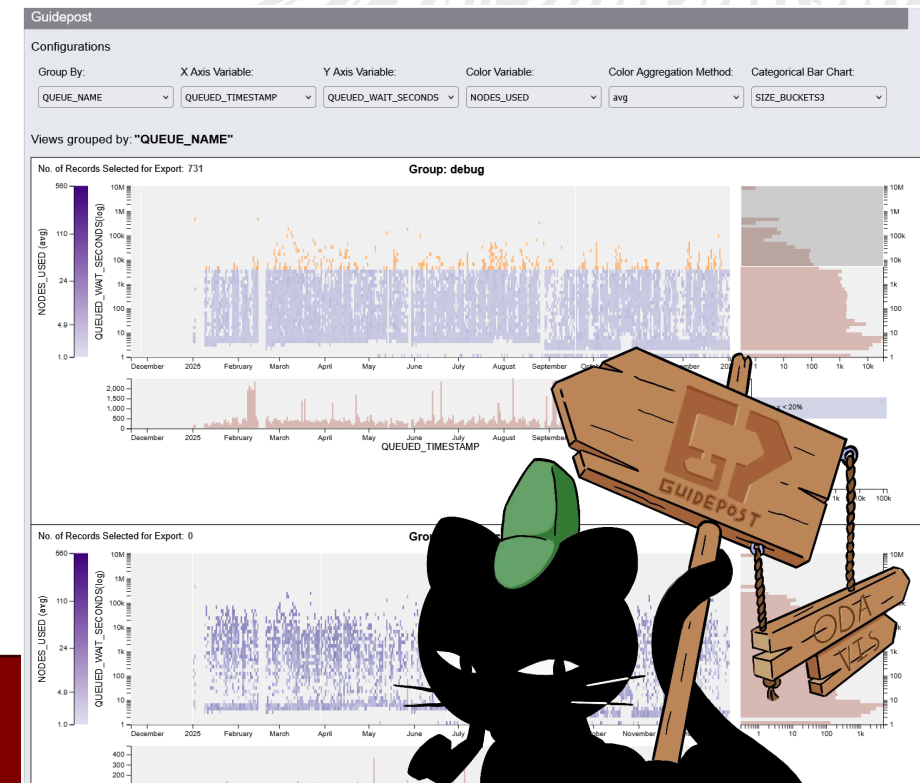
And many, many more...

Successful integration of LLM workflows into HPC analysis will require multiple interventions to address diverse issues related to deployment and fidelity.

Connor Scully-Allison

✉ cscully@uchicago.edu

🐙 <https://github.com/cscully-allison>



```
$ pip install guidepost
```

Publicly available

```
$ pip install guidepost
```



Argonne
NATIONAL LABORATORY

NATIONAL
LABORATORY
OF THE ROCKIES

Lawrence Livermore
National Laboratory

RECENTLY UPDATED WITH
NEW FUNCTIONALITY!

COMPATIBLE WITH ALL
HISTORICAL JOBS DATA.



Related Papers

- [1] **Scully-Allison, Connor**, Ian Lumsden, Katy Williams, Jesse Bartels, Michela Taufer, Stephanie Brink, Abhinav Bhatele, Olga Pearce, and Katherine E. Isaacs. “Design Concerns for Integrated Scripting and Interactive Visualization in Notebook Environments.” *IEEE Transactions on Visualization and Computer Graphics* 30, no. 9 (September 2024): 6572–85. <https://doi.org/10.1109/TVCG.2024.3354561>.
- [2] **Scully-Allison, Connor**, Katy Williams, Stephanie Brink, Olga Pearce, and Katherine E. Isaacs. “A Tale of Two Models: Understanding Data Workers’ Internal and External Representations of Complex Data.” arXiv, January 21, 2025. <https://doi.org/10.48550/arXiv.2501.09862>.
- [3] Brink, Stephanie, Ian Lumsden, **Connor Scully-Allison**, Katy Williams, Olga Pearce, Todd Gamblin, Michela Taufer, Katherine E. Isaacs, and Abhinav Bhatele. “Usability and Performance Improvements in Hatchet.” In *2020 IEEE/ACM International Workshop on HPC User Support Tools (HUST) and Workshop on Programming and Performance Visualization Tools (ProTools)*, 49–58, 2020. <https://doi.org/10.1109/HUSTProtools51951.2020.00013>.
- [4] Menear, Kevin, **Connor Scully-Allison**, and Dmitry Duplyakin. “Quantifying Uncertainty in HPC Job Queue Time Predictions.” In *Practice and Experience in Advanced Research Computing 2024: Human Powered Computing*, 1–3. PEARC ’24. New York, NY, USA: Association for Computing Machinery, 2024. <https://doi.org/10.1145/3626203.3670627>.
- Selected Other Publications:
- [5] **Scully-Allison, Connor**, and Katherine E. Isaacs. “Evaluating Communication Pattern Representations in Execution Trace Gantt Charts.” In *2024 IEEE Working Conference on Software Visualization (VISSOFT)*, 1–11, 2024. <https://doi.org/10.1109/VISSOFT64034.2024.00011>.
- [6] Brink, Stephanie, Michael McKinsey, David Boehme, **Connor Scully-Allison**, Ian Lumsden, Daryl Hawkins, Treece Burgess, et al. “Thicket: Seeing the Performance Experiment Forest for the Individual Run Trees.” In *Proceedings of the 32nd International Symposium on High-Performance Parallel and Distributed Computing*, 281–93. HPDC ’23. New York, NY, USA: Association for Computing Machinery, 2023. <https://doi.org/10.1145/3588195.3592989>.