

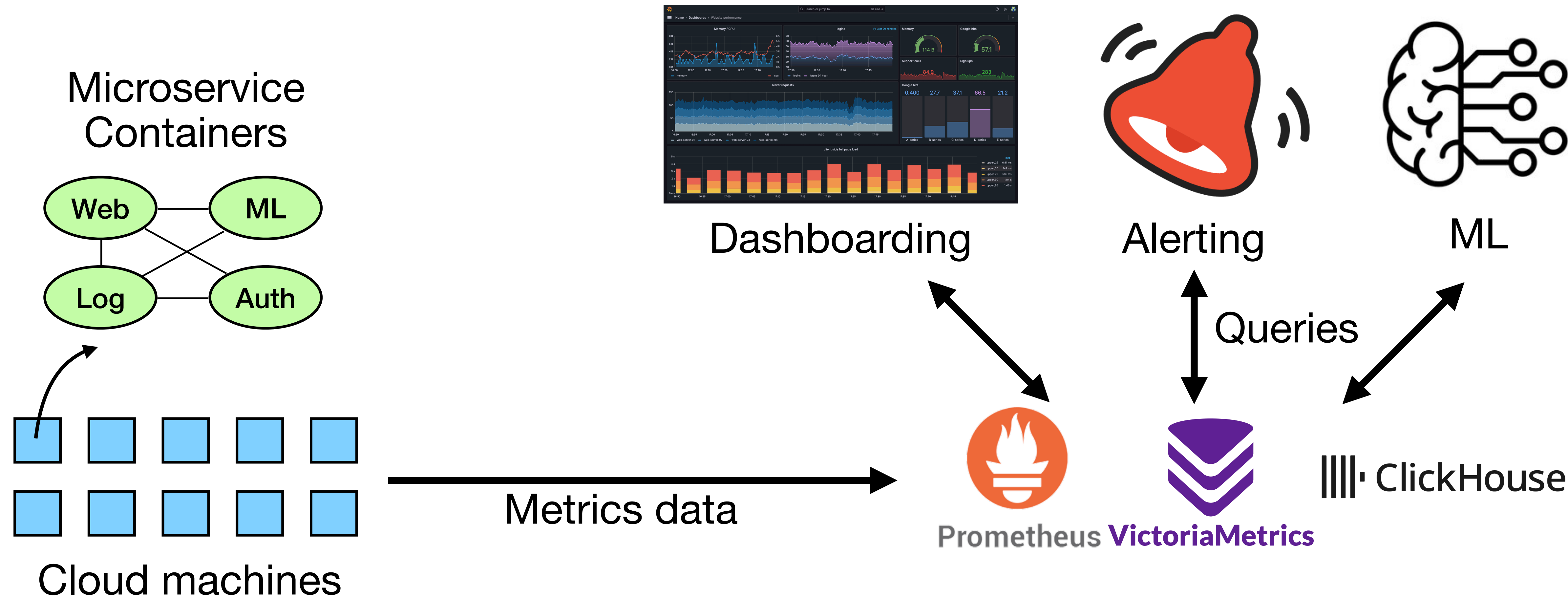
Reducing Cloud Costs for Metrics Observability by 50x



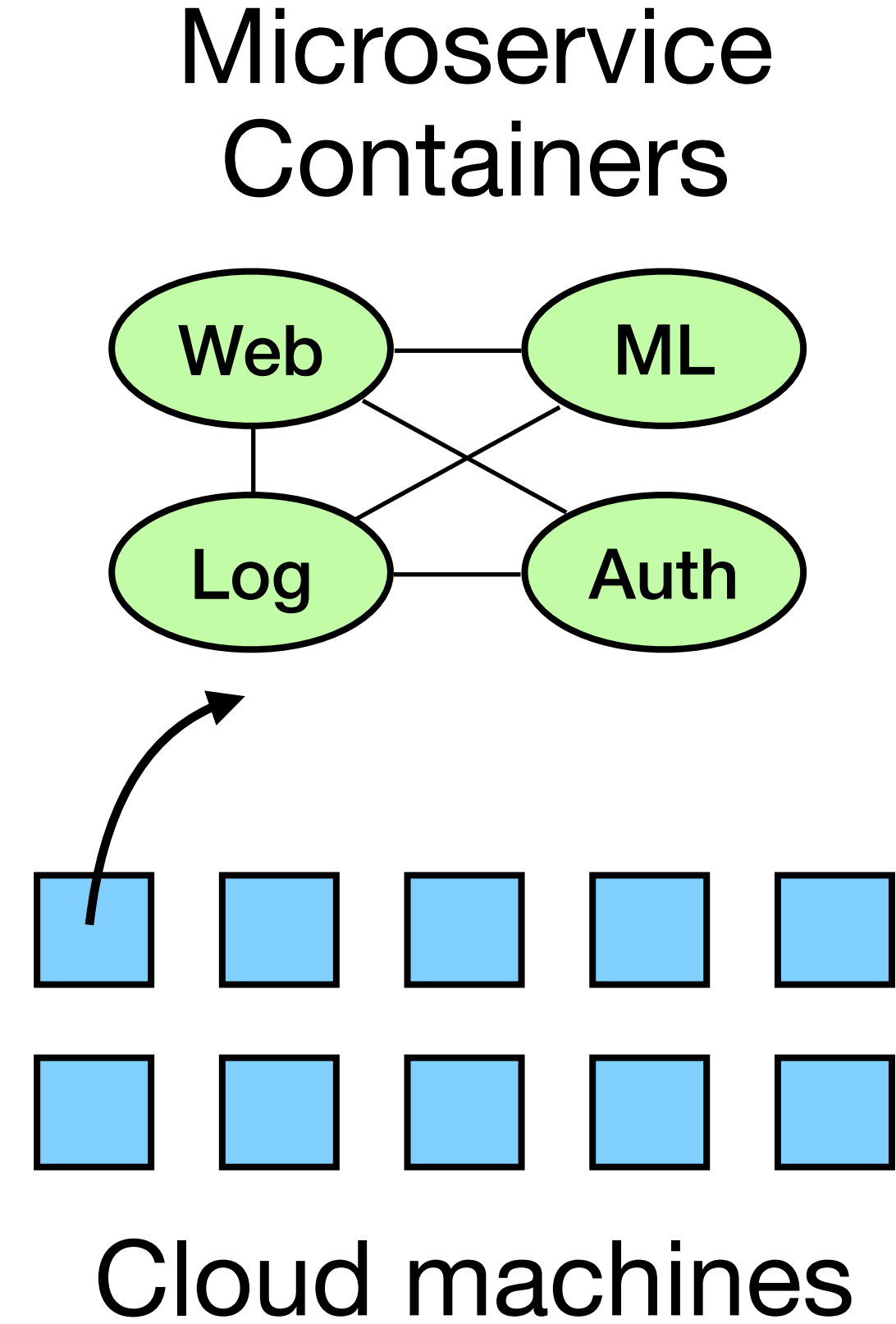
Milind Srivastava, Alan Liu, Vyas Sekar

Carnegie Mellon University, University of Maryland

Need for Metrics Observability



Example: Monitoring Cloud Machines



Time	CPU	hostID	container	
			ID	Type
5:00:00	10%	Host1	c1	Web
5:00:00	35%	Host1	c2	Web
5:00:00	15%	Host1	c3	Auth
5:00:00	90%	Host2	c4	ML
...				
5:00:05	15%	Host1		Log
5:00:05	75%	Host2		Web
...				

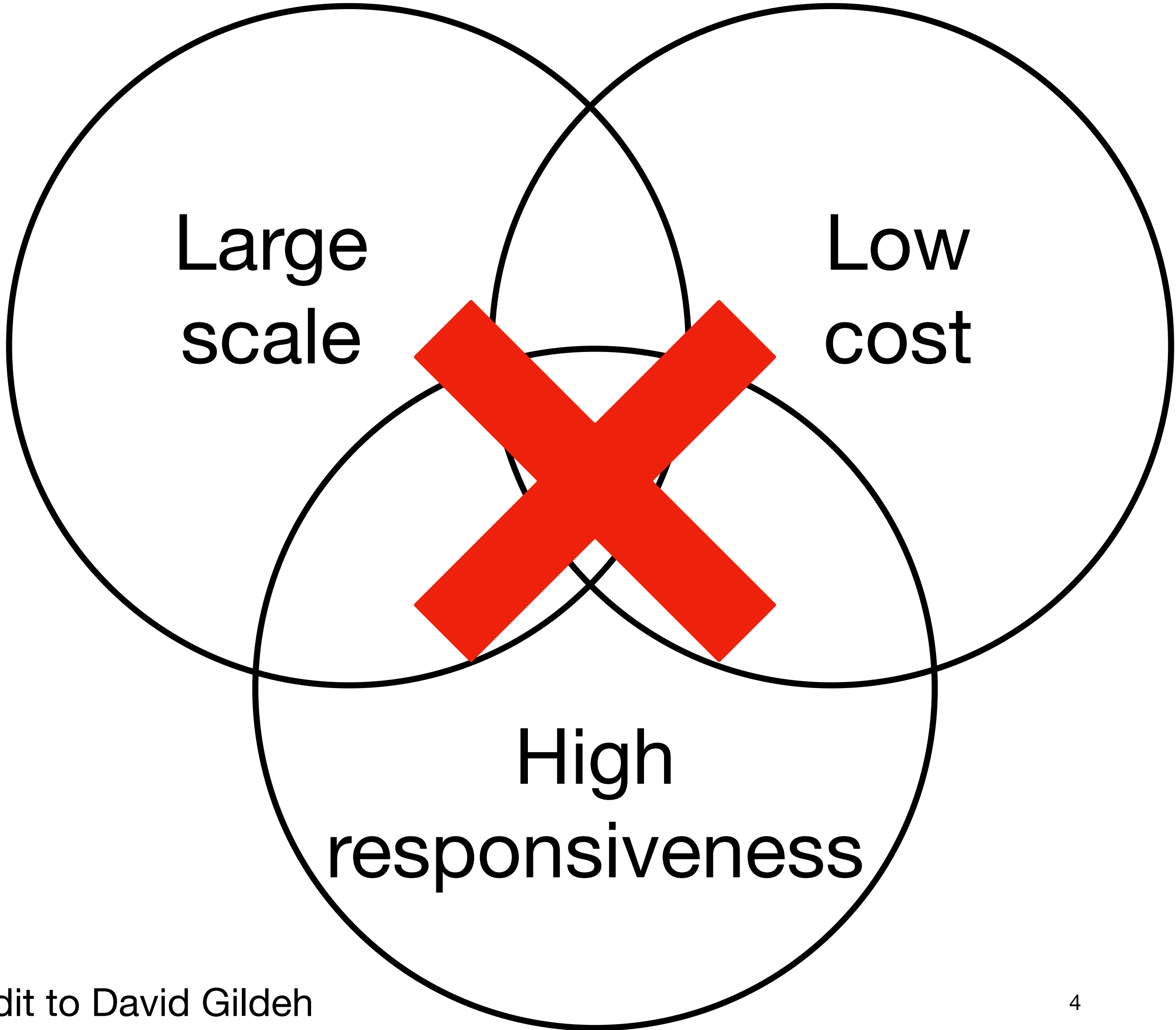
Metrics data



For the last 5 min:
For each machine:
Avg(CPU)



Observability's CAP Theorem



Datadog's \$65M Bill and Why Developers Should Care

Observability bills can surprise companies, and developers may be on the hook for changing code if bills grow too large.

May 17th, 2023 10:56am by [Loraine Lawson](#)

Monitorama PDX 2023 - How to Scale Observability Without Bankrupting the Company - David Gildeh, Netflix - From the talk description: "Every company has struggled with controlling their Observability costs as the amount of data that needs to be collected, stored and queried in real time has gone up exponentially! Netflix has had to solve this problem as it rapidly scaled into a web scale

36%
> 1\$M per year

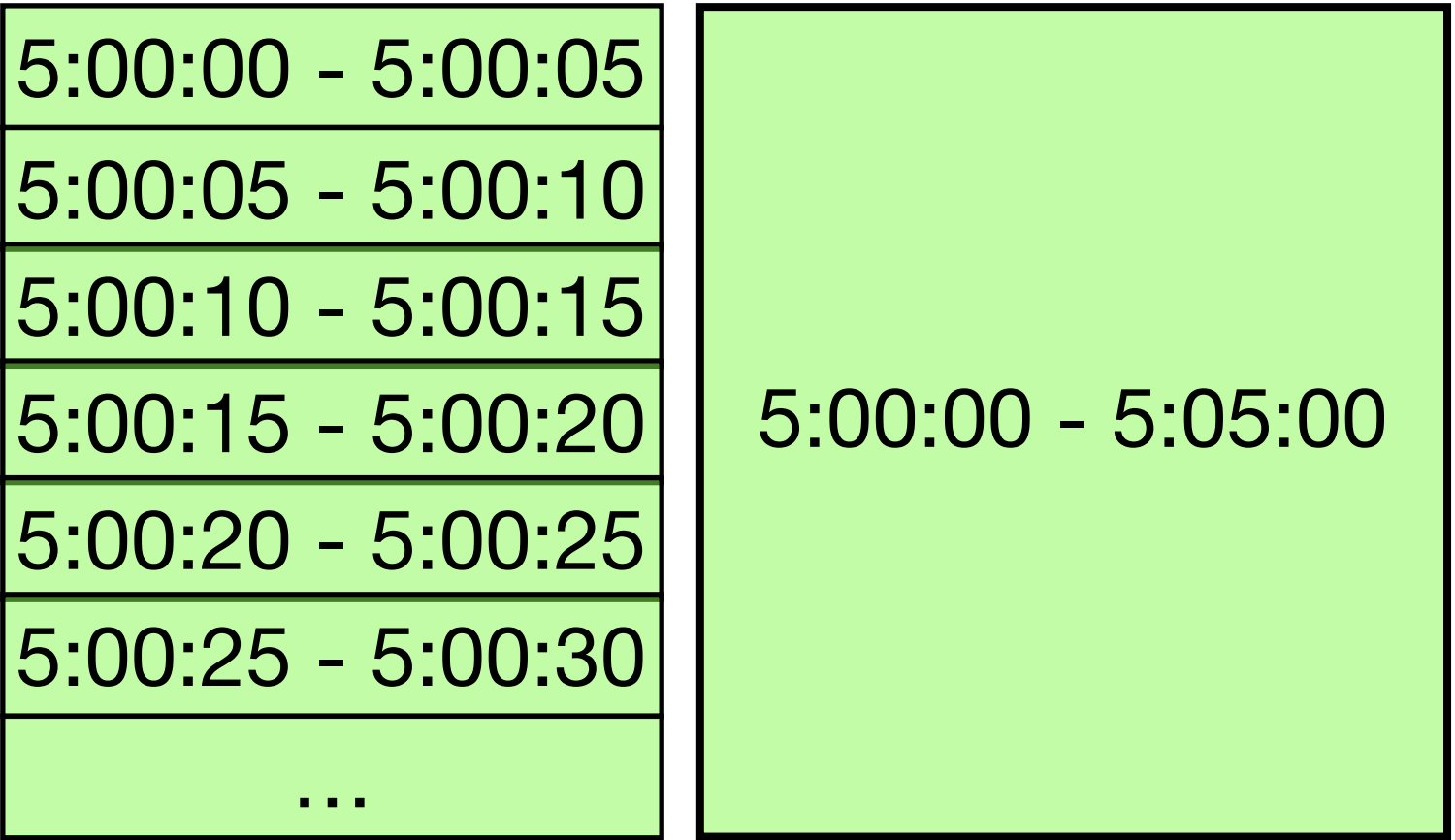
4%
> \$10M per year

Gartner report

Status Quo is Useful but Incremental

Time	CPU	hostID	container	
			ID	Type
5:00:00	10%	Host1	c1	Web
5:00:00	35%	Host1	c2	Web
5:00:00	15%	Host1	c3	Auth
5:00:00	90%	Host2	c4	ML
...				
5:00:05	15%	Host1		Log
5:00:05	75%	Host2		Web
...				

Sample/Drop data



Rollup data

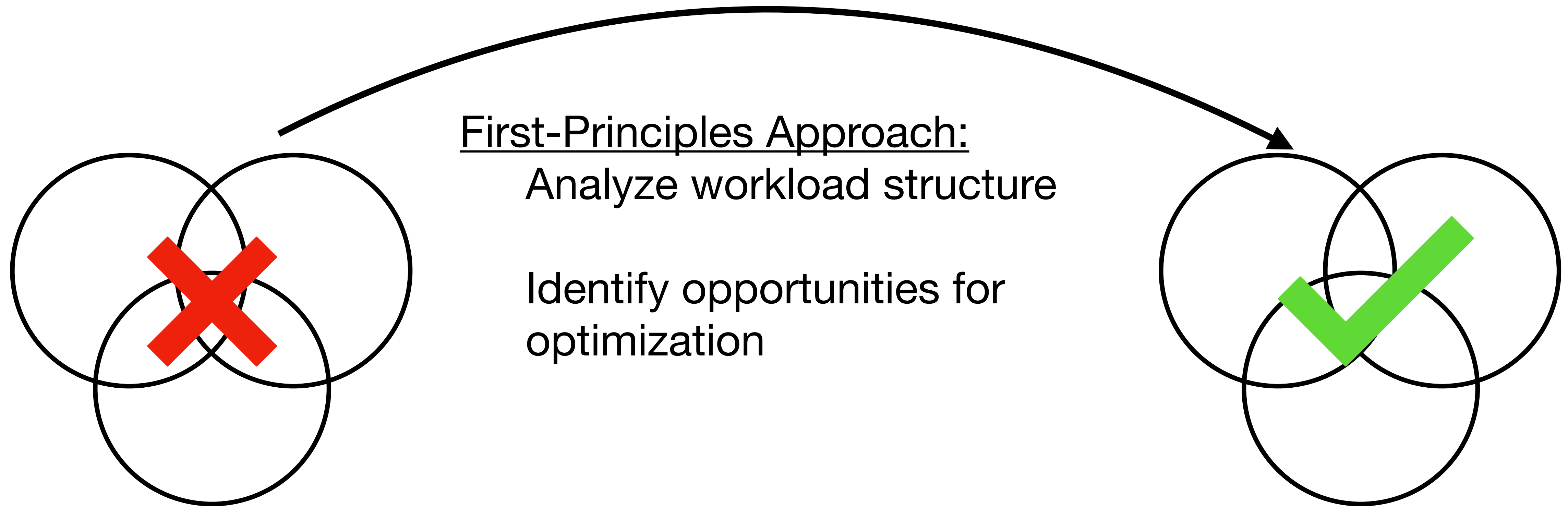
Lindorm TSDB: A Cloud-native Time-series Database for Large-scale Monitoring Systems

Gorilla: A Fast, Scalable, In-Memory Time Series Database

Engineering
M3: Uber’s Open Source, Large-scale Metrics Platform for Prometheus

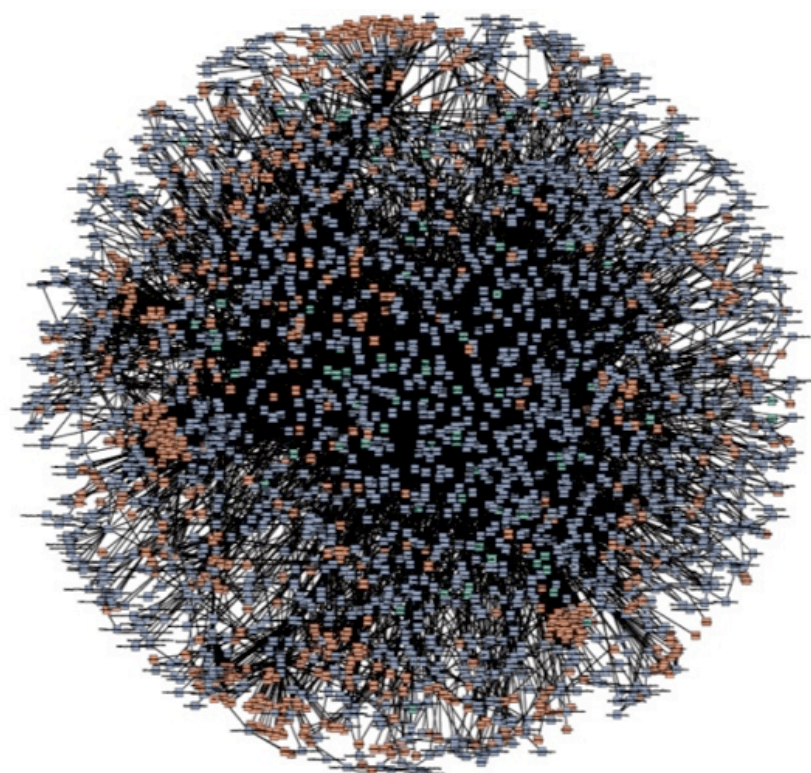
Faster TSDBs

We Need a Quantum Leap



Challenges in Metrics Observability

engine handles around 2,500 queries per second (Figure 1), about 8.5 billion data points per second (Figure 2), and approximately 35 Gbps (Figure 3). These numbers have been constantly trending upwards at a rate much higher than Uber's organic growth due to the increased adoption of metrics across various parts of our stack.



High volume data

Time	CPU	hostID	container	
			ID	Type
5:00:00	10%	Host1	c1	Web
5:00:00	35%	Host1	c2	Web
5:00:00	15%	Host1	c3	Auth
5:00:00	90%	Host2	c4	ML
...				
5:00:05	15%	Host1		Log
5:00:05	75%	Host2		Web
...				

High cardinality multi-dimensional data

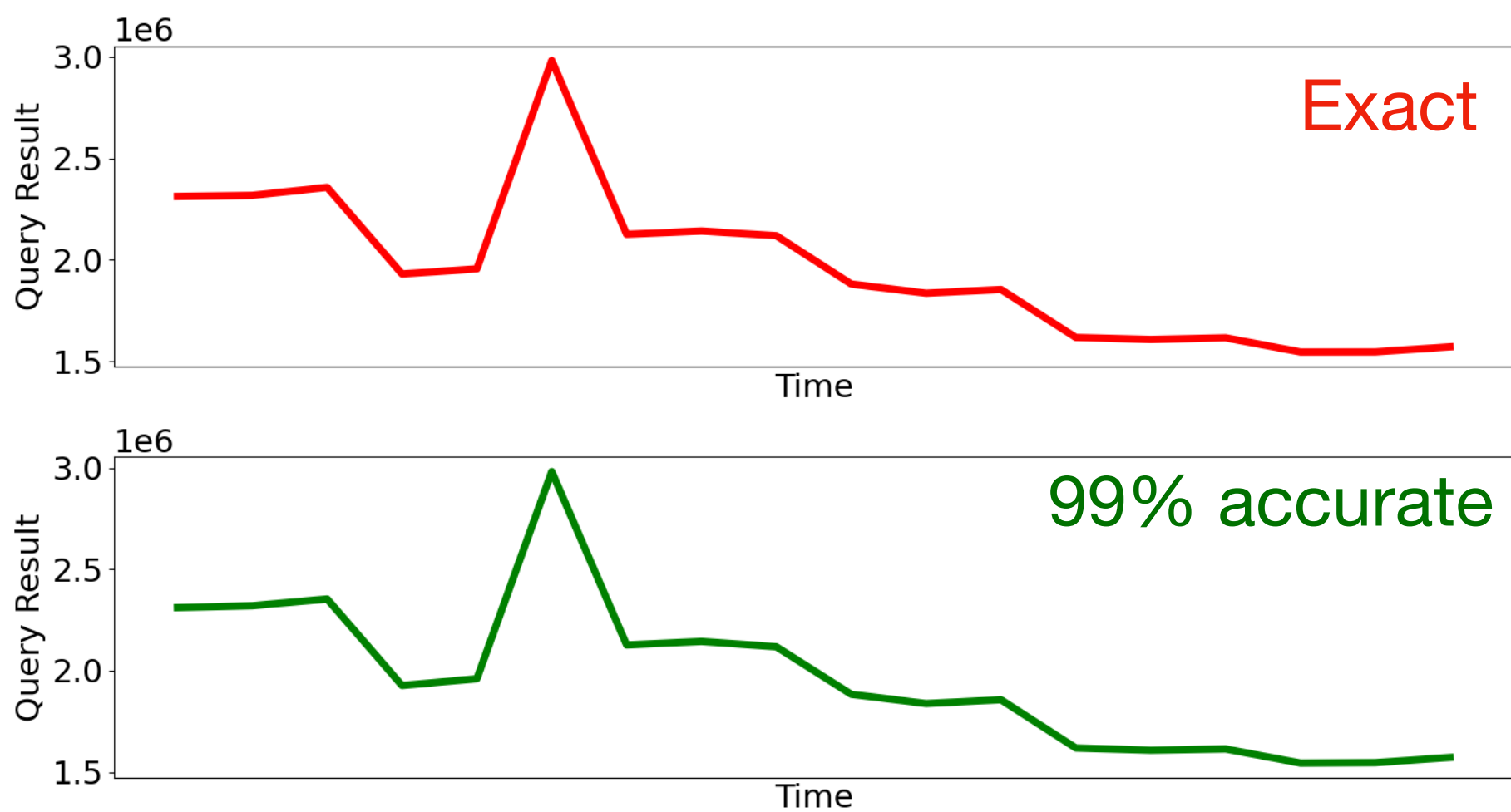
For the last 5 min:
For each hostID:
Avg(CPU)



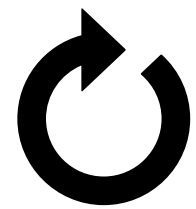
Spatio-temporal aggregates on high cardinality multi-dimensional data

Structural Analysis of Metrics Observability Workloads

Spot the difference?



Aggregates are often
approximatable

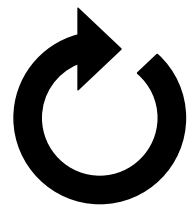


every 15 sec

For the last 5 min:
For each hostID:
Avg(CPU)

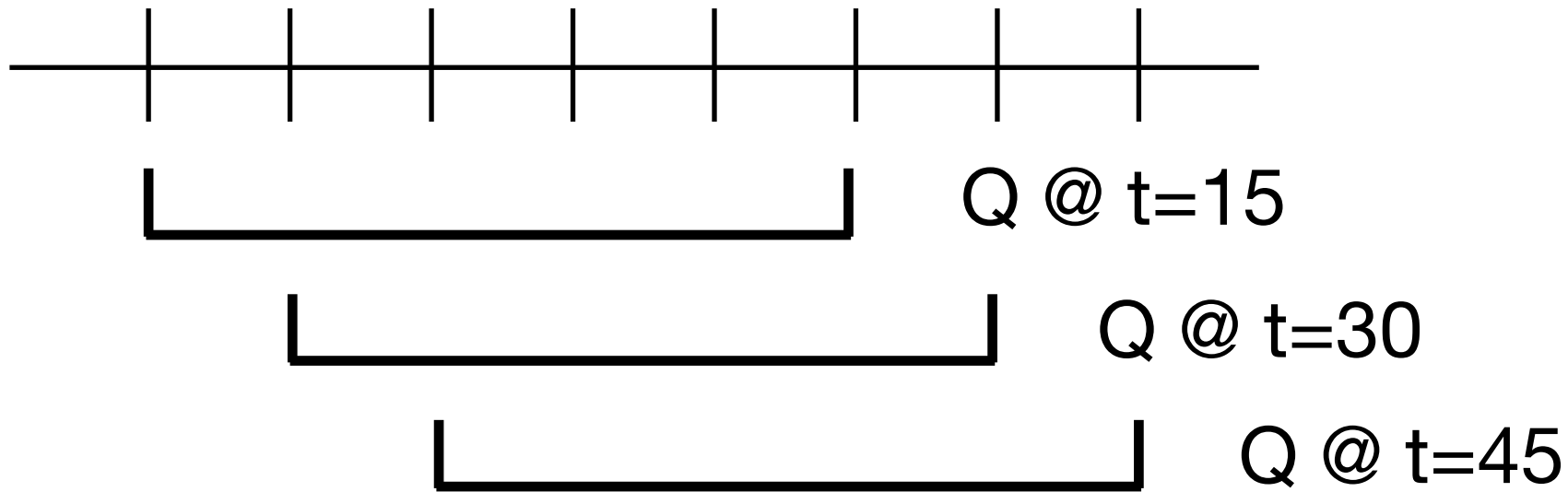


Queries are
predictable



every 15 sec

For the last 5 min:
For each hostID:
Avg(CPU)



Sliding window
queries

Reimagining the talks from Day 1:

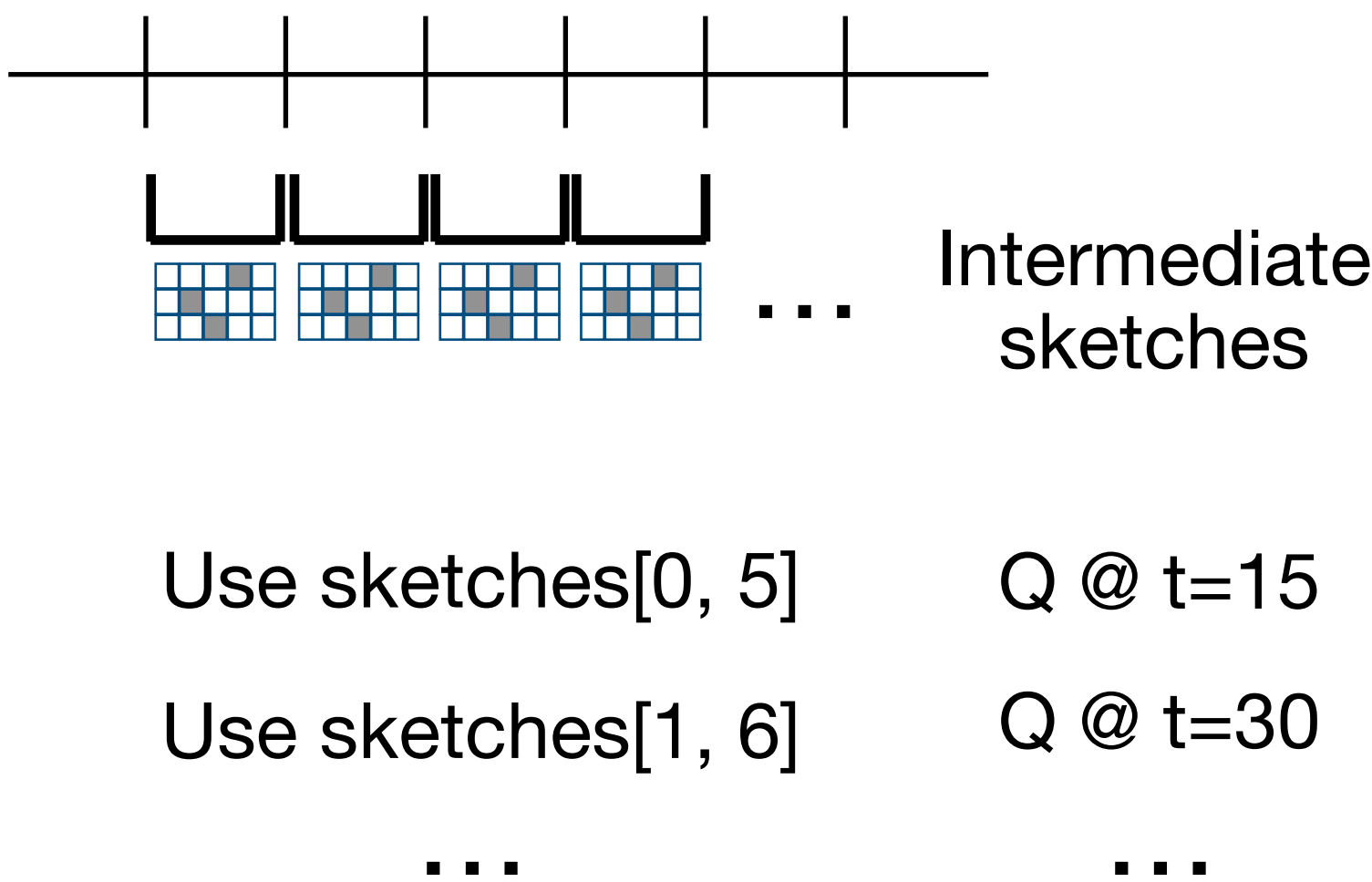
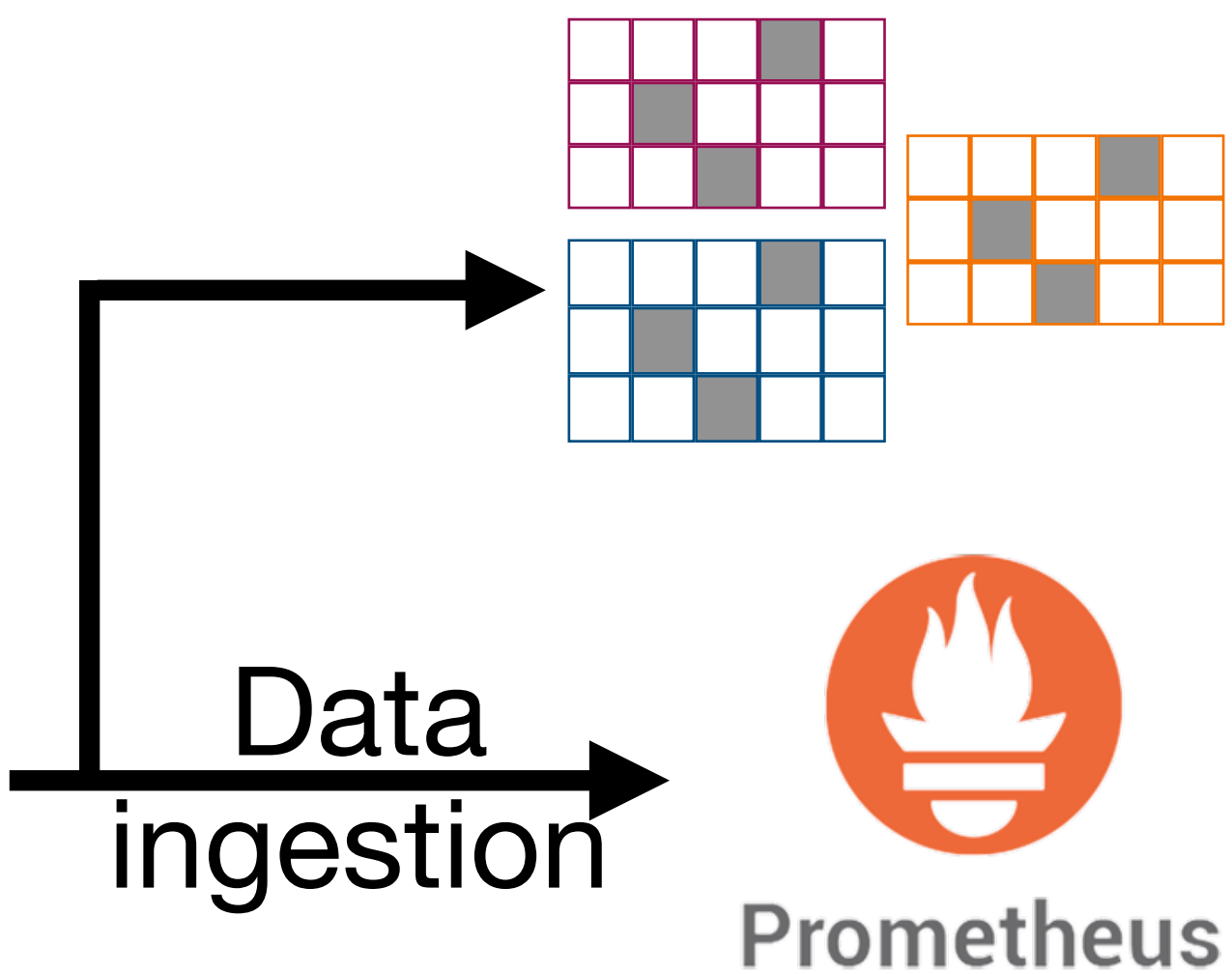
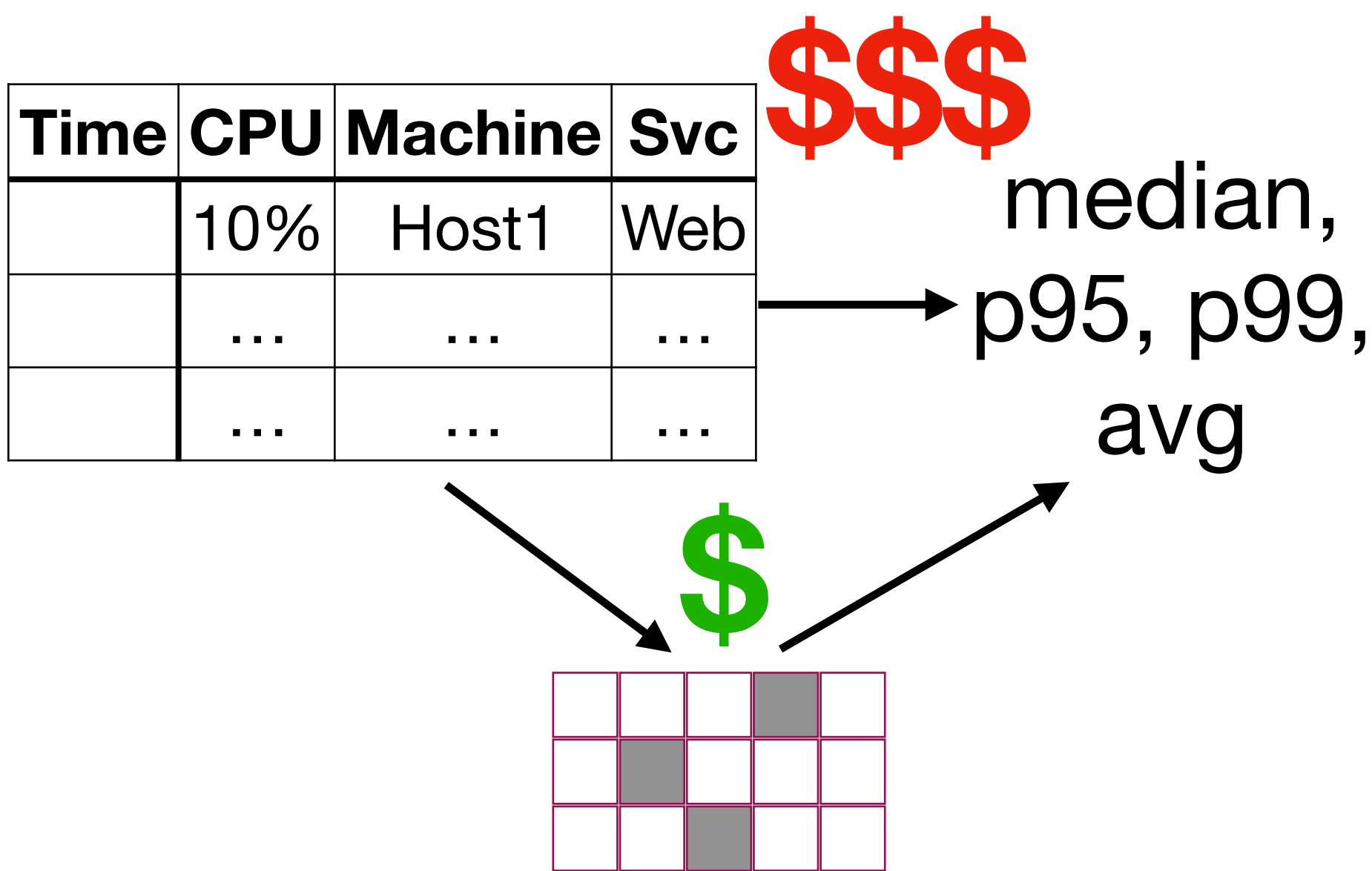
- John: Jobs ask for $F(X)$, ask them what $F(X)$ quality they are willing to accept! More degrees of freedom for analytics tasks
- Prashant: Carbon($F(X)$, 100% fidelity) vs Carbon ($F(X)$, 95% fidelity), more elasticity in utility
- Jayaram: Memory is a pain, but forgetting past is OK —> Make approximate forgetting a first class citizen
- Indy: Sloppy is OK — Make sloppy a first class citizen
- ...

Our Structure-Aware Insights

Sketch-based aggregation

Precompute sketches

Incremental computation



High cardinality multi-dimensional data

Aggregates are often approximatable

Queries are predictable

Sliding window queries

Requirements to Synthesize Insights into a Design

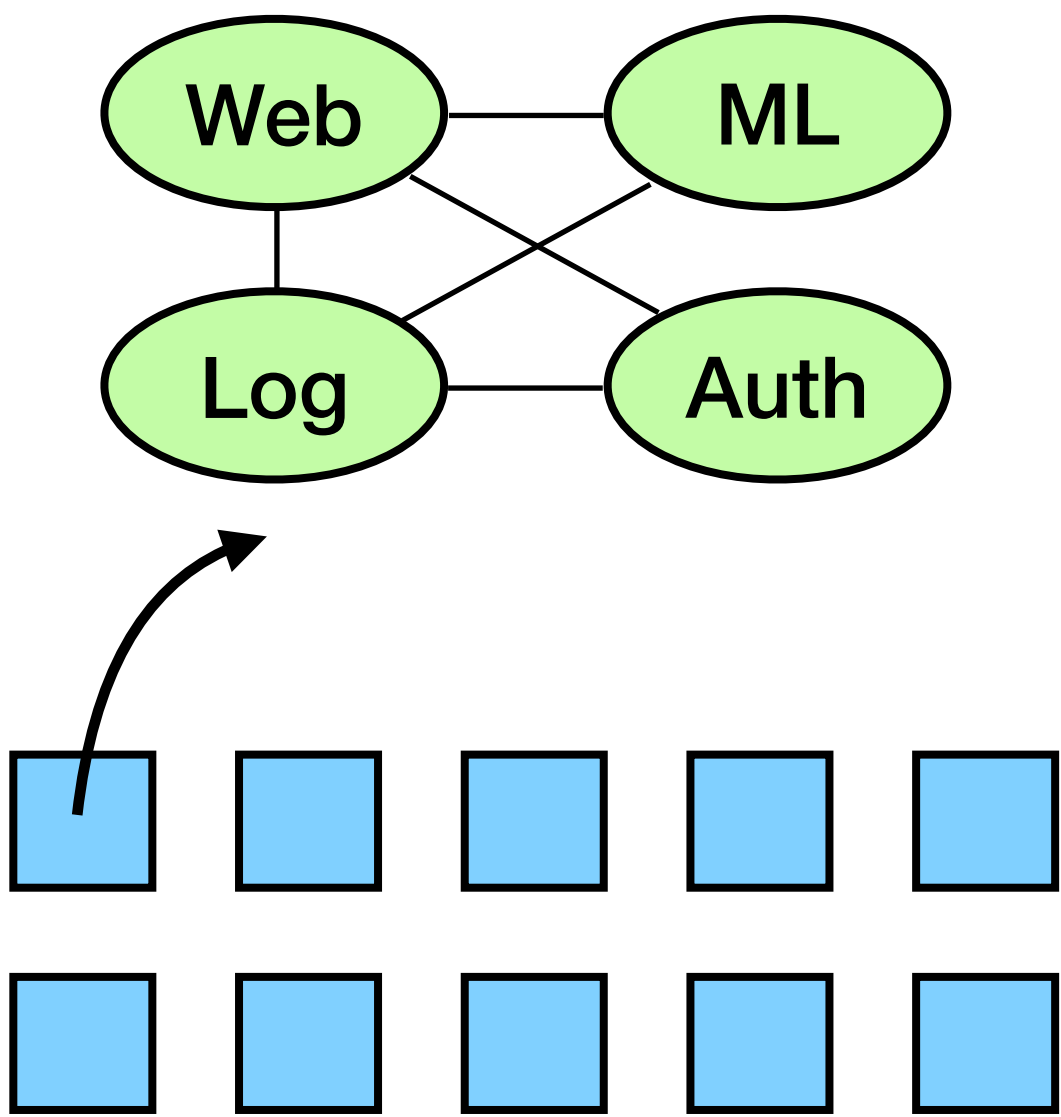
Coverage over our insights

Low development + deployment effort

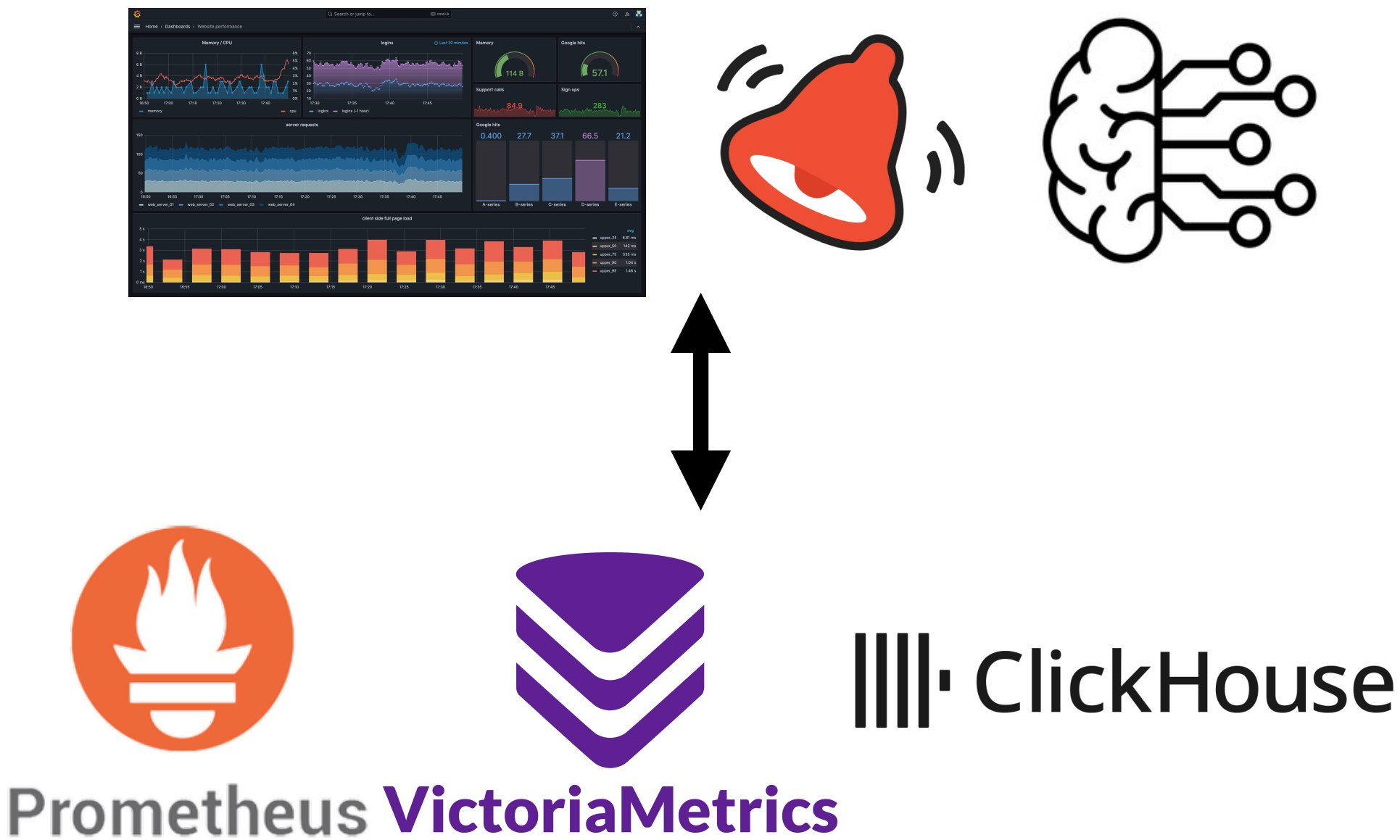
“Can this design implement insight XYZ”

“How easy is to implement insight XYZ”

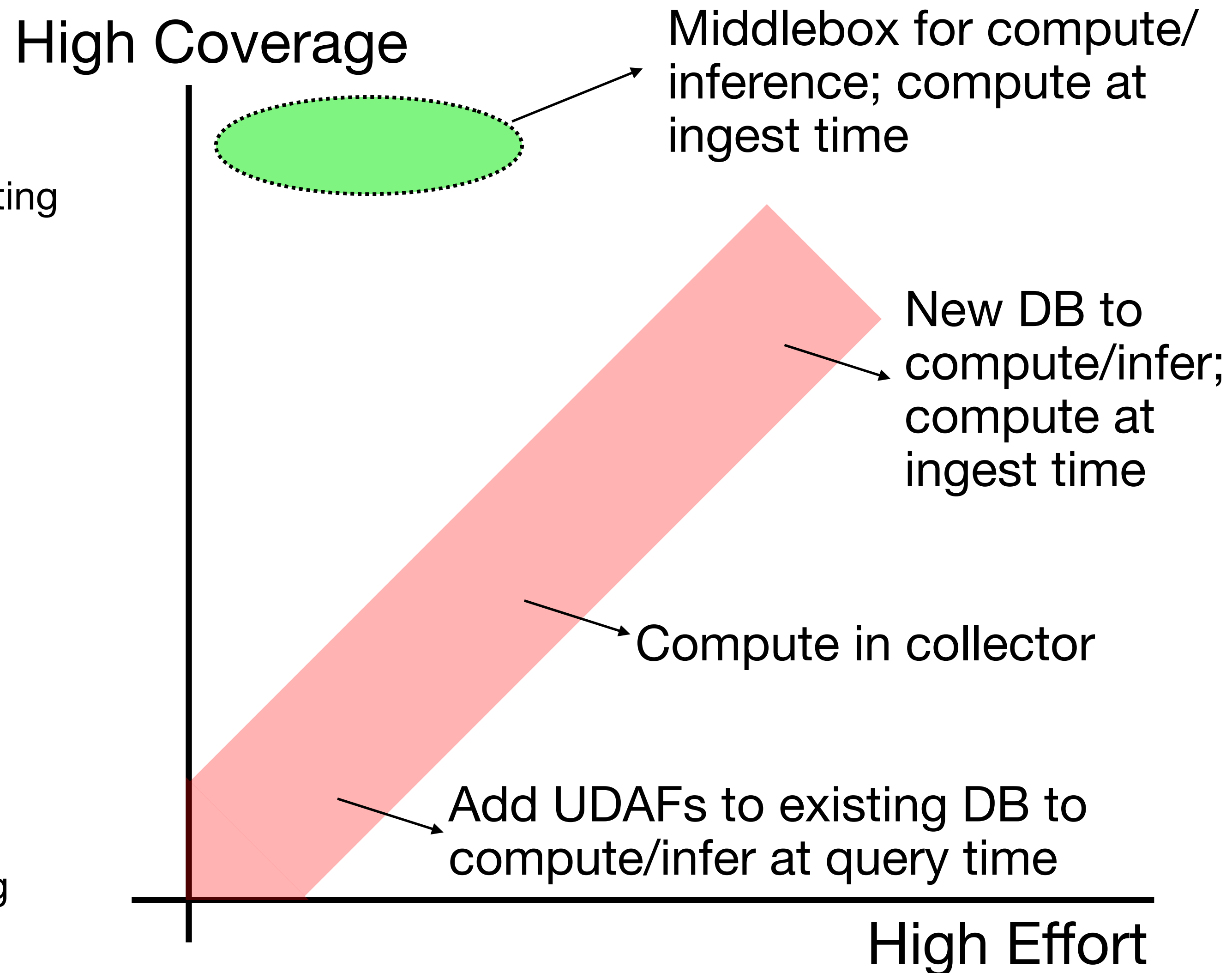
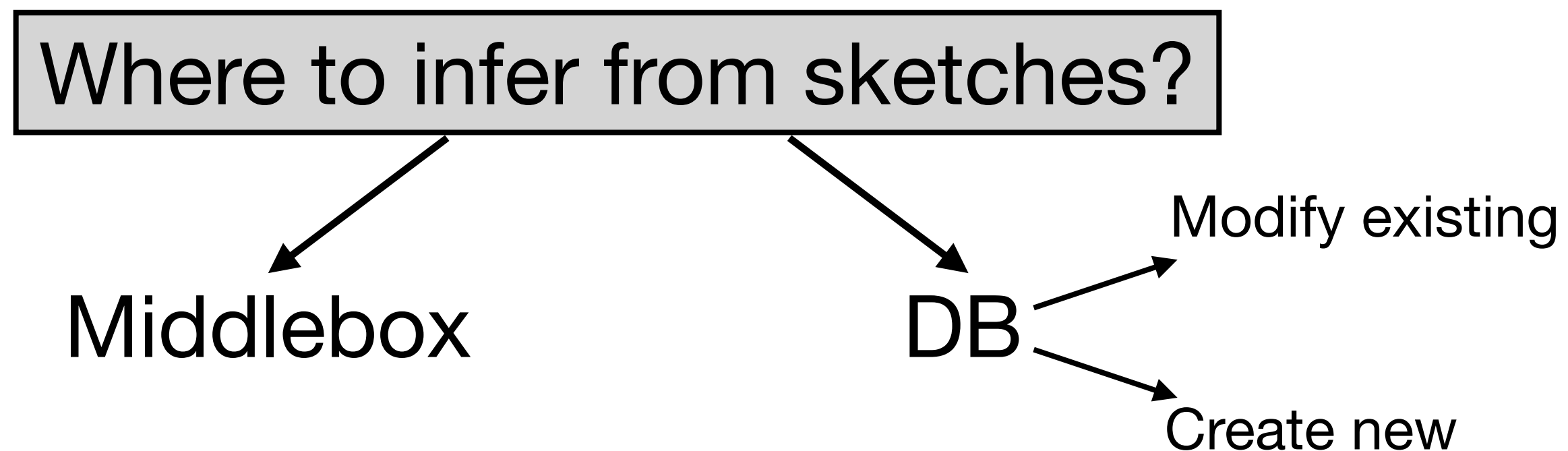
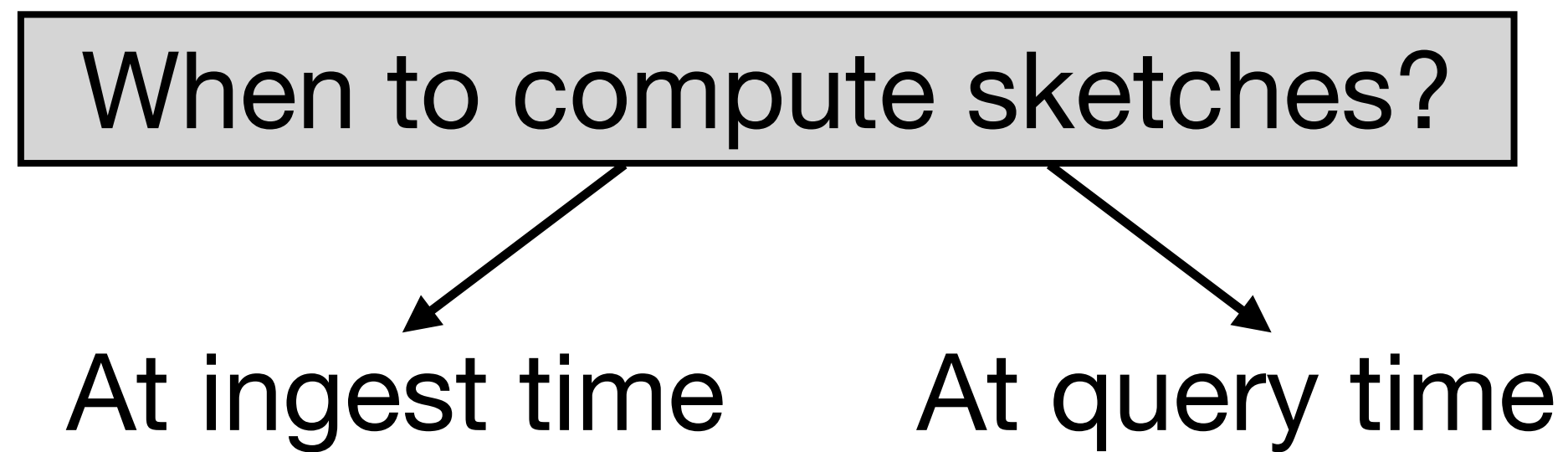
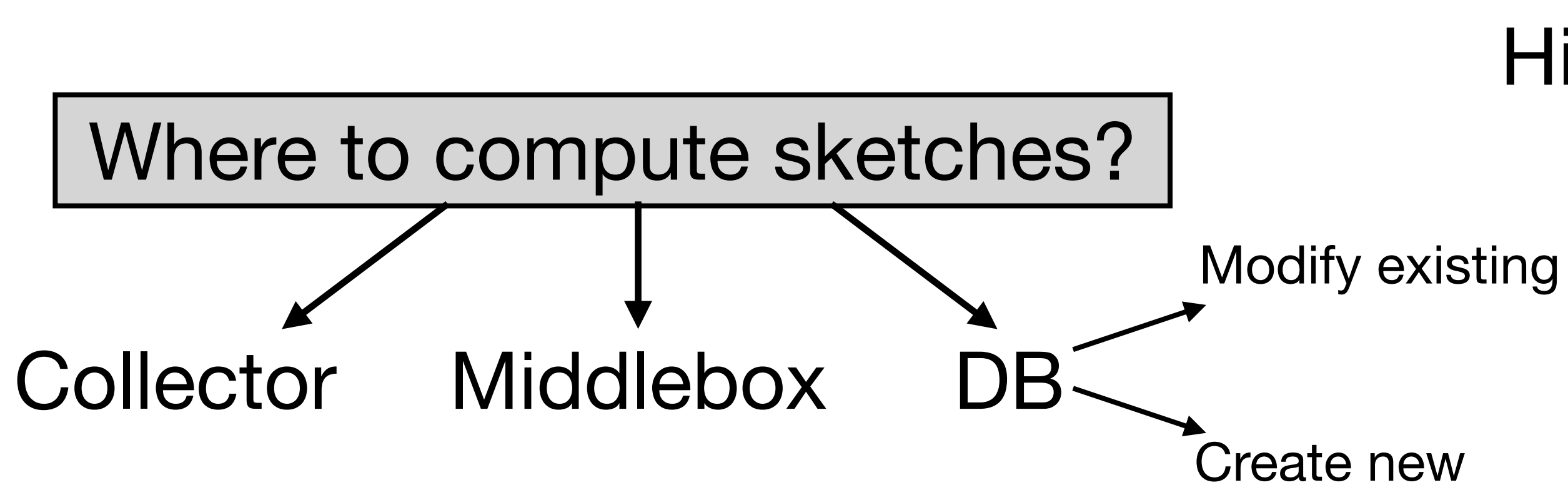
“How easy is to retrofit this in an existing deployment”



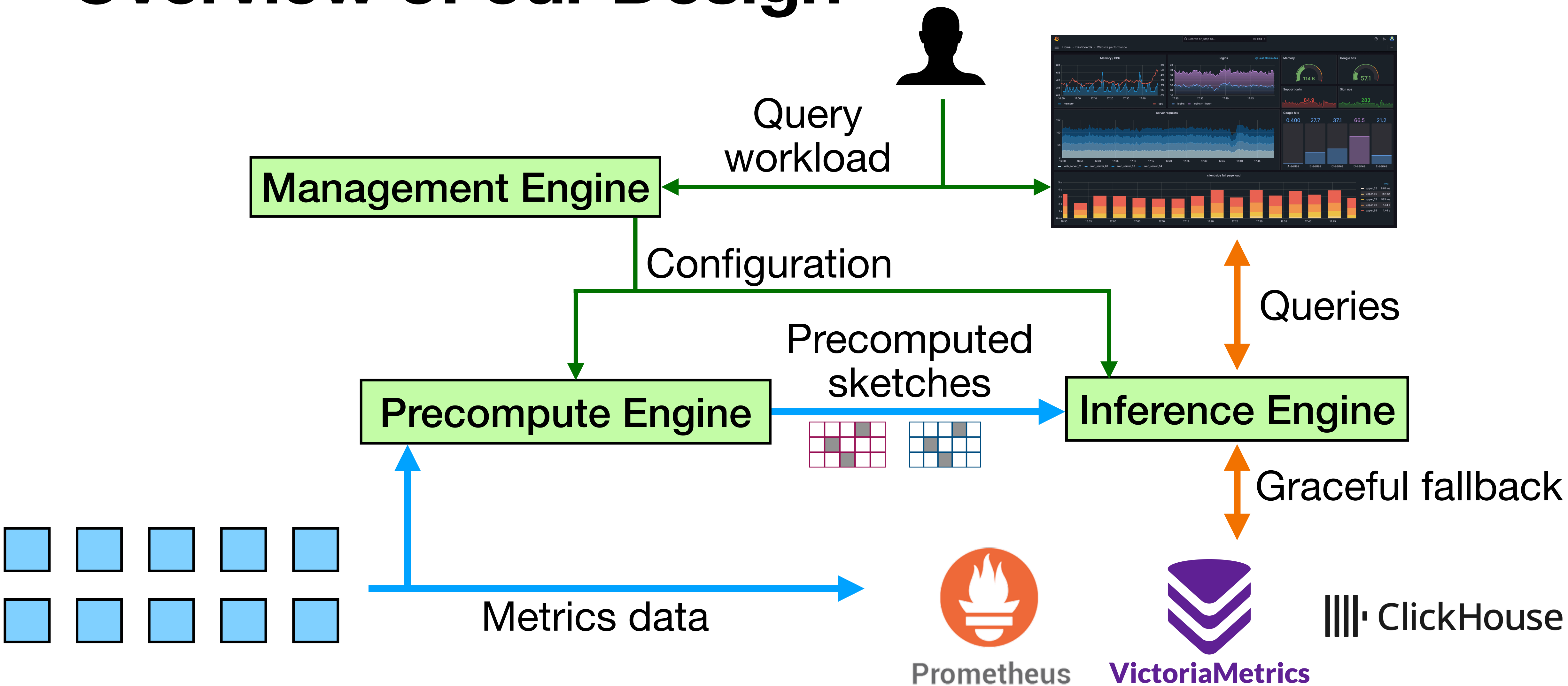
Metrics data



Naive Designs Fail to Meet Requirements

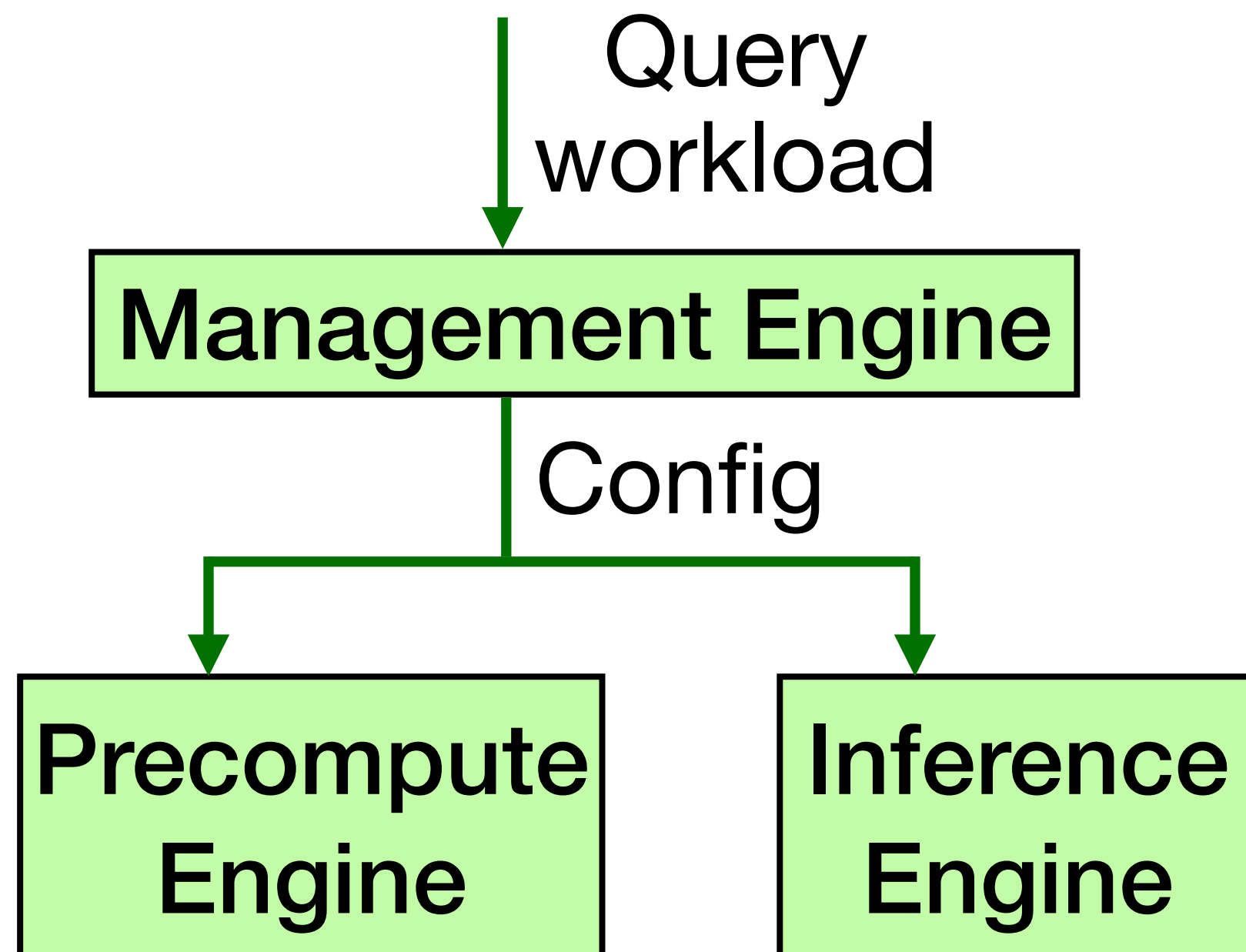


Overview of our Design



Legend: **Data**; **Query**; **Config**

Design Contribution: Configuration Language



Config language requirements:

Realizable enough

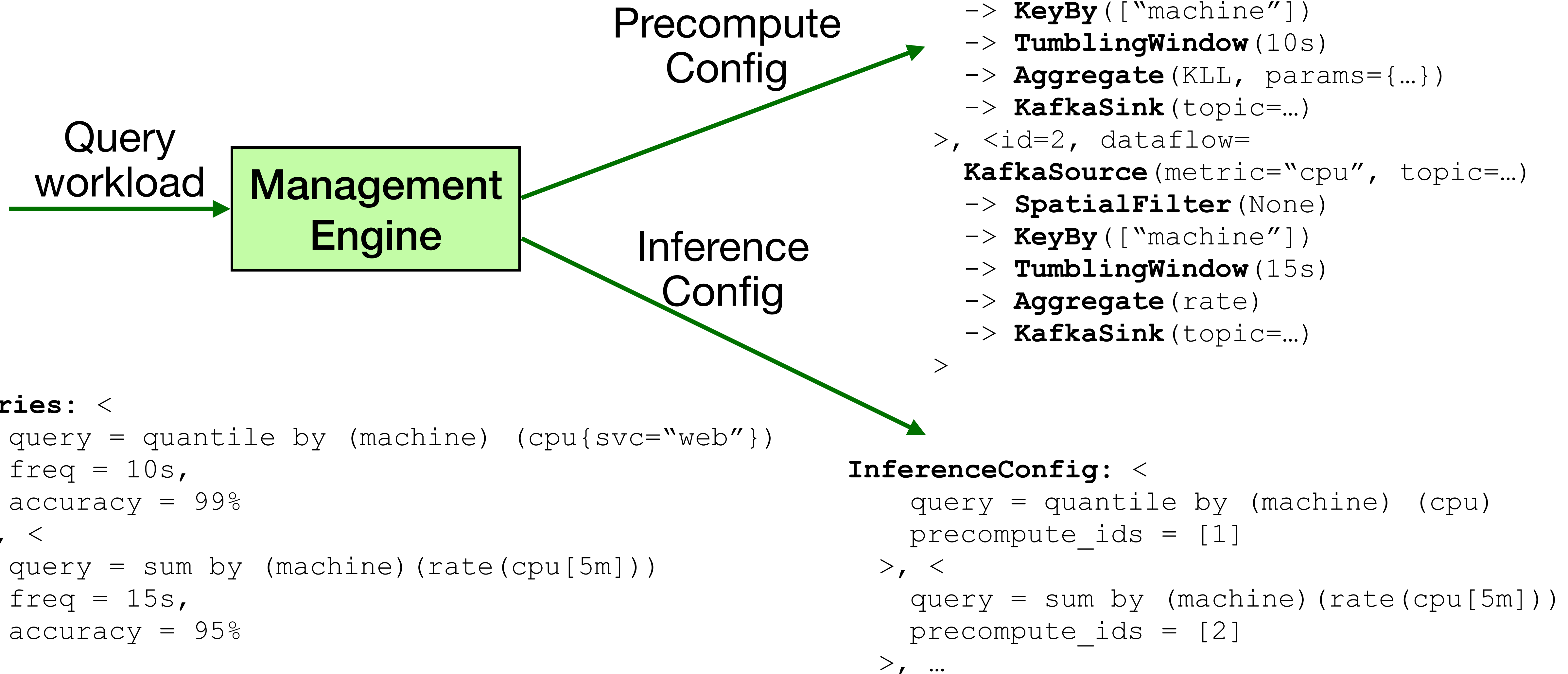
Extensible enough

Side goal: Configs should be easy to realize using off-the-shelf components

PrecomputeConfig: List of dataflow graph, each with a restricted set of operations

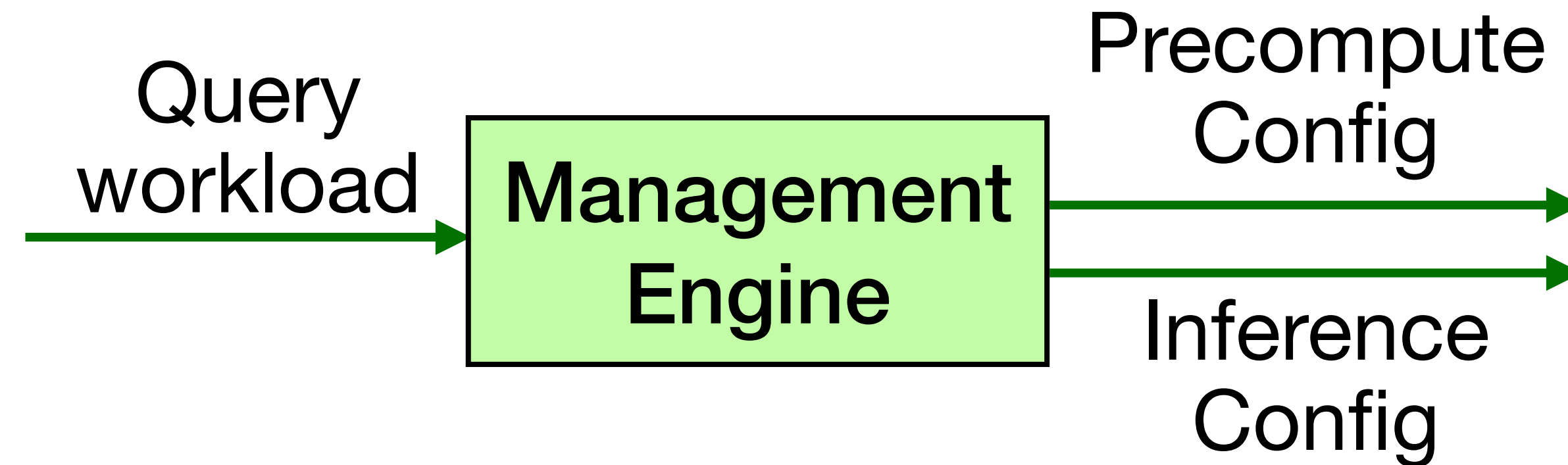
InferenceConfig: Map of each query to a list of dataflows to use

Example Configs



Legend: **Data**; **Query**; **Config**

Design Contribution: Management Engine



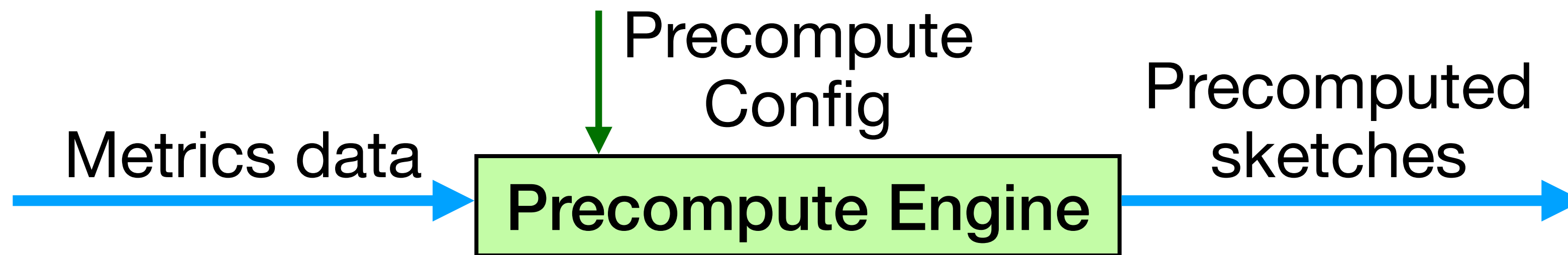
- Metrics of success:
- Coverage over queries
 - Extensibility to new queries & sketches

Naive design: Bespoke per-query logic to derive configs

Key insights:

- (a) Diverse spatio-temporal aggregations can be clustered into a few buckets
- (b) Treat each query independently, then combine configs

Design Contribution: Precompute Engine



Metrics of success:

- Low overhead
- Extensible to other sketches

Metric:

```
cpu = 25%
timestamp = 5:00:00
labels = <
  machine = host1
  svc = web-server
  containerID = c239
>
```

PrecomputedAggregate:

```
time_window = [10, 20)
subpopulation = <
  svc = web-server
>
id = 1
precompute = CountSketch(

|   |   |   |
|---|---|---|
| 5 | 2 | 1 |
| 8 | 0 | 1 |

...)
```

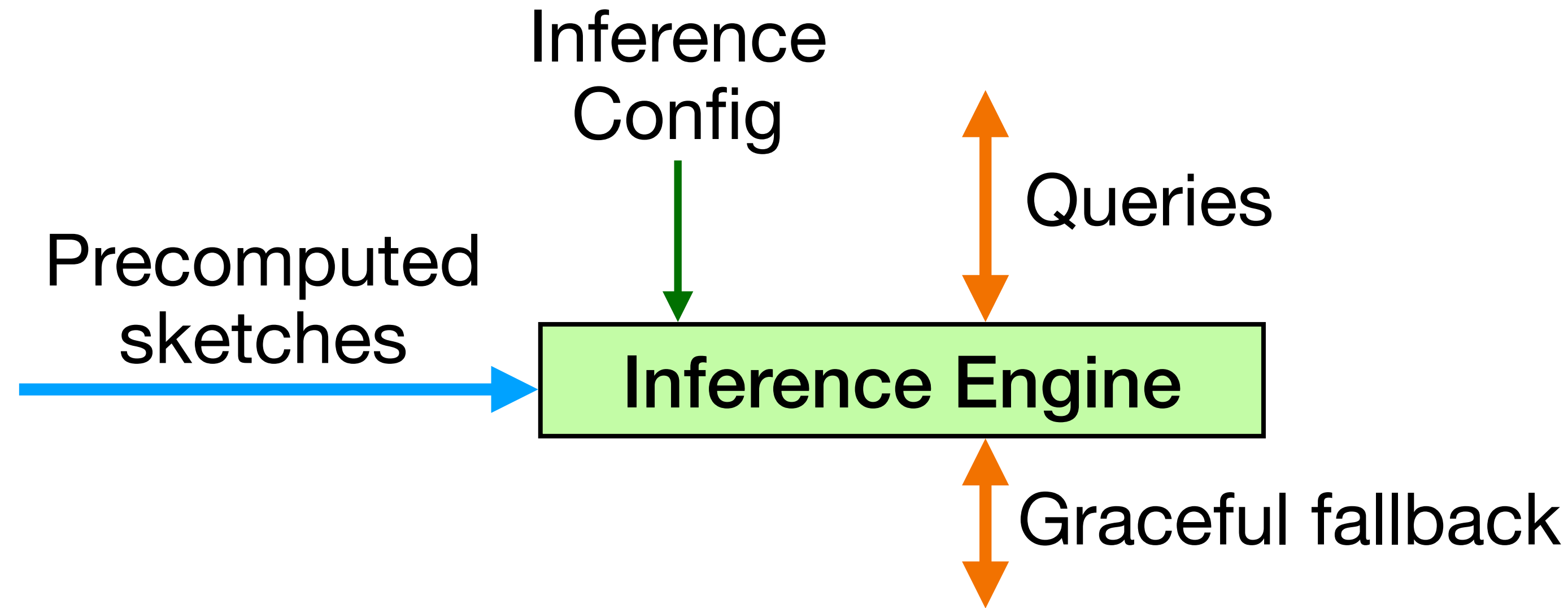
Naive design: Implement from scratch

Key insight: Use off-the-shelf stream processing engines.

- Abstractions well-suited for implementing the Precompute Config.
- Enough leeway for extensibility and sketch-specific optimizations (SketchLib, Sketchovsky, Nitrosketch, etc.)

Legend: **Data**; **Query**; **Config**

Design Contribution: Inference Engine



Metrics of success:

- Low overhead
- Extensible to new queries
- Graceful fallback

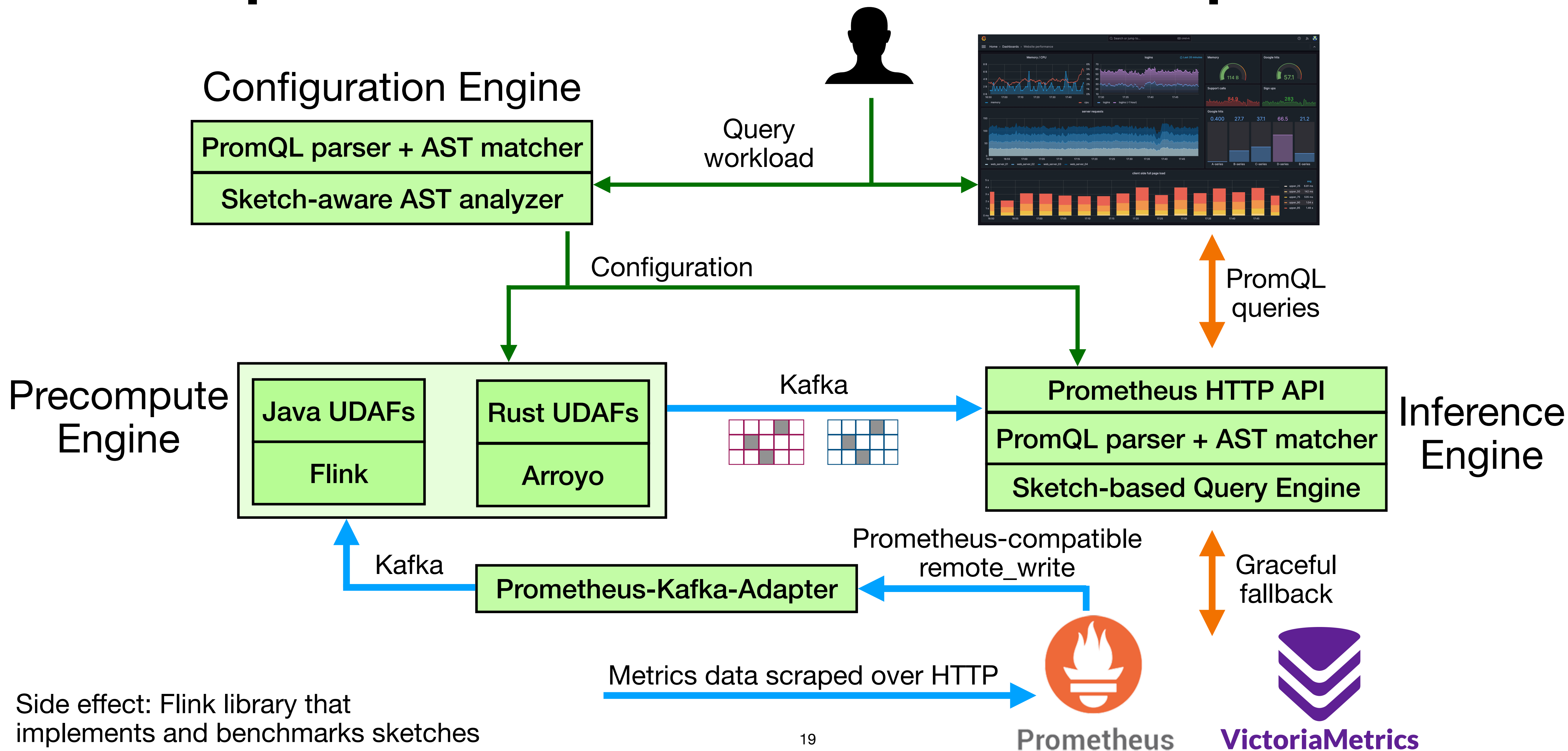
Naive design: Bespoke per-query logic

Key insight:

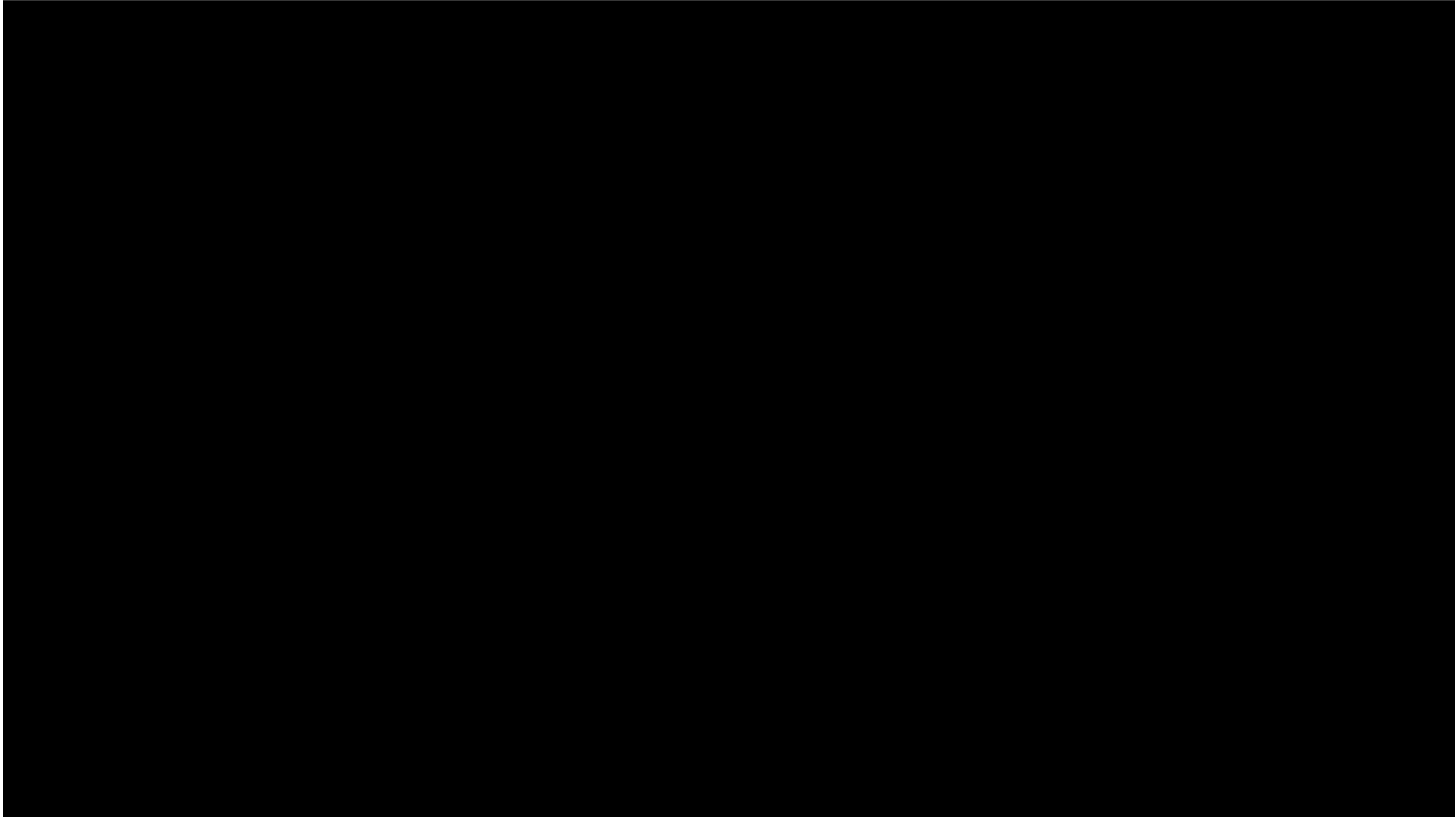
- (a) Query inference logic exhibits a narrow set of patterns
- (b) Leverage data access patterns in queries e.g. store sketches in a ring buffer, faster query inference for “subtractable” sketches

Legend: **Data**; **Query**; **Config**

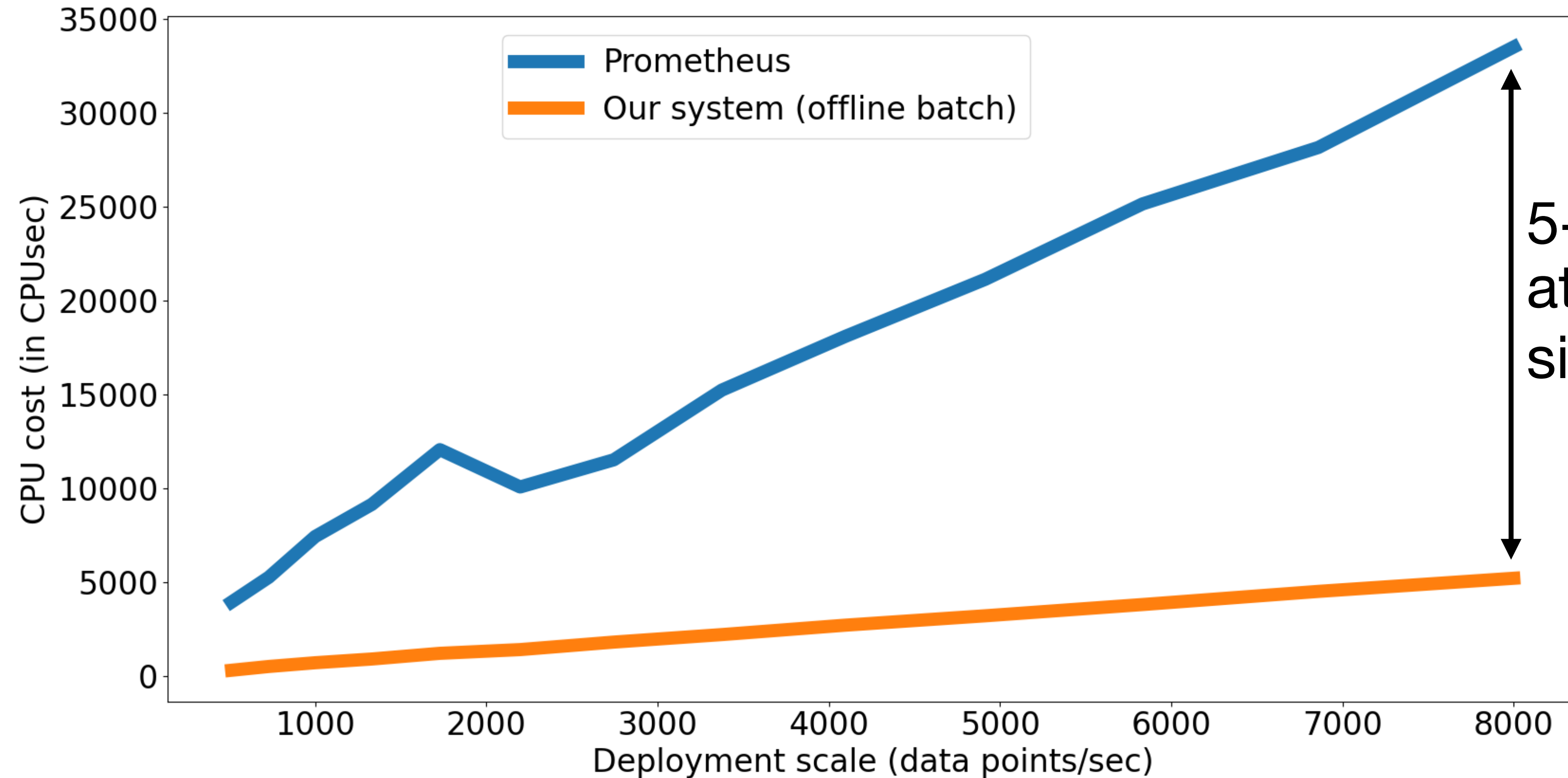
An Implementation for Prometheus-compatible DBs



It's Demo Time!



There is a Pot of Gold



5-6x benefit
at tiny scales with
simple queries

Uber ~ 9B data points/sec

Alibaba cluster traces ~
1M data points/sec

Query functions - sum, increase, rate

The Road Ahead: The Narrow Waist of Analytics

