Aalto University
School of Science

Monitoring, Observability, and Experimenting for Machine Learning Systems

Hong-Linh Truong
Department of Computer Science
linh.truong@aalto.fi, https://rdsea.github.io

CS-E4660 Advanced Topics in Software Systems, Fall 2024
18/09/2024

Learning objectives

- Able to analyze the role of measurement, monitoring and observability in real-world cases for R3E
- Understand and develop methods with key steps and important tools for monitoring, observability and experimenting
- Able to apply these methods for ML systems

The role of measurement, monitoring and observability

Development vs Runtime activities

Design, test and benchmark R3E

- analyze R3E for individual components
- analyze/model complex dependencies
- design logs, metrics and traces for capturing states and complex dependencies

Monitoring/observability and runtime adaptation

- runtime monitoring and observability
- states, performance and failure analytics
- runtime controls (constraints, rules, actions)

Measurement, monitoring, and observability for R3E (1)

- **Instrumentation**
  - insert probes into systems to measure system behaviors directly or produce logs
- **Sampling**
  - use components to sample system behaviors
- **Profiling**
  - statistically measures time, usage, frequency, and duration
- **Tracing**
  - record trails of calls/flows in details
- **Automatic vs manual**
  - compiler, wrapper, interceptors, etc. vs manual, or combination of both

Measurement, monitoring, and observability for R3E (2)

- **Monitoring and Tracing**
  - perform sampling or instrumentation to collect and share metrics, logs, traces
  - visualize what has been happened
- **Observability**
  - evaluate and interpret measurements for specific contexts
  - understand and explain the systems states, dependencies, etc.

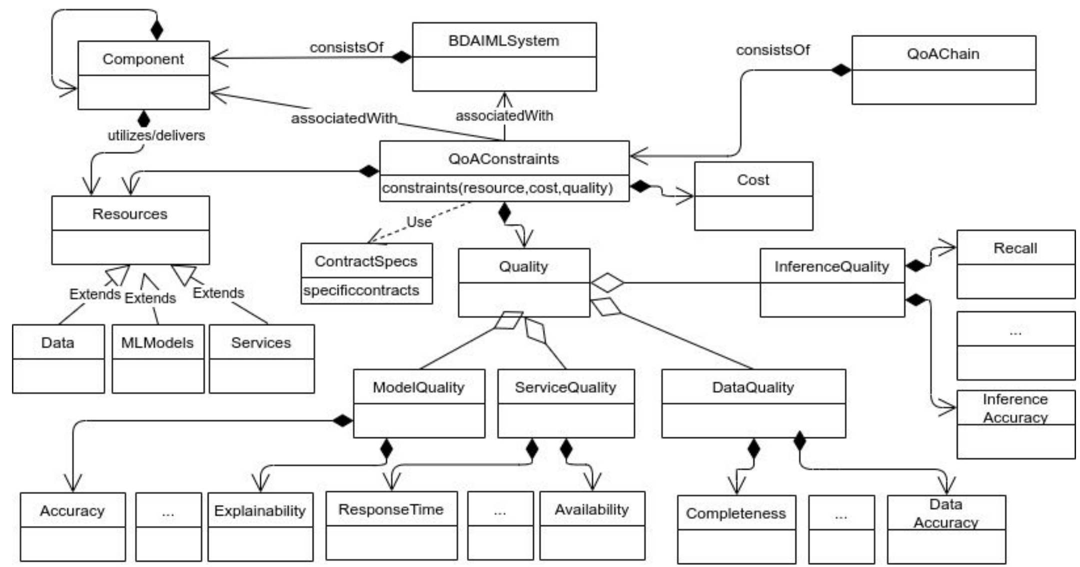
Recall: strongly interdependencies

Any problem would lead to a huge waste (engineering effort, operation cost, societal impact due to wrong inference/prediction)

as a process
(e.g. feature engineering, training process)

as an artefact
(e.g. trained model)

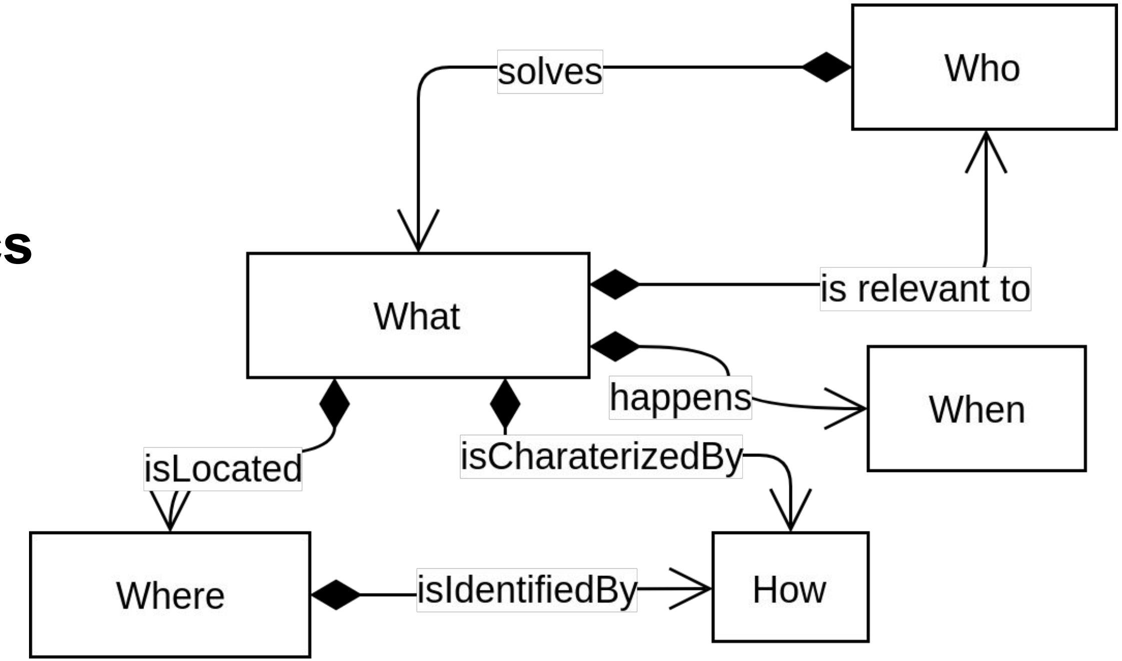
as a service
(e.g., a deployed ML prediction)



Methods

What/Which, Where, When, Who and How

Understand W4H aspects for analytics of ML systems



Key steps – What/Which

- **Understand and identify indicators/metrics characterizing your systems**
- **Common metrics vs specific ML applications/systems**
  - not just about ML models or inference
  - different relevance/importance based on specific contexts
- **Most critical problems are due to complex dependencies that are not common**
  - root cause analysis will be tricky
- **For which purposes?**
  - Site Reliability Engineering (SRE, https://sre.google/), Test-Driven Development (TDD), explainability

Key steps – Where and When

- **Where: as a “ space” dimension**
  - tightly coupled or isolated/loosely coupled
  - different places
    - software/system layers, components and systems boundaries
    - dependencies among components
    - development/configuration pipelines
- **When: as a „time“ dimension**
  - design, test/training, or runtime (DevOps)
  - further divided into sub states

Key steps - How

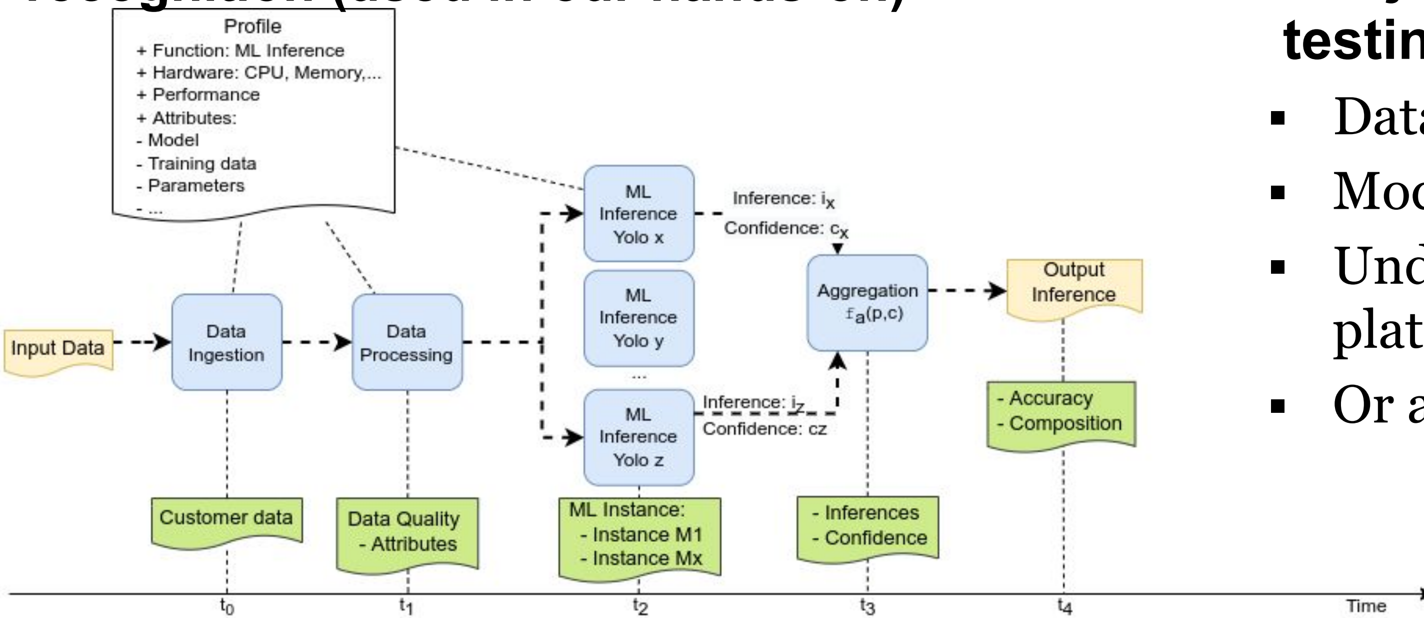
- **Characterize dependencies among components**
  - understand the system as a whole
  - can focus on identified critical parts
  - include also development processes, data, software artefacts and execution environments
- **Select tools for capturing metrics**
- **Understand what kind of changes/designs we must do**
- **Do monitoring and analysis**
- **Integrate many types of data for monitoring and observability**

Apply W4H for benchmarking, monitoring, validation and experimenting

- **Determines clearly system boundaries**
  - the system under study, the system used to judge, and the environment
  - “domain-driven/oriented” and bounded context principles
- **Understands dependencies**
  - among components in distributed big data/ML systems in distributed computing platforms
  - single layer as well as cross-layered dependencies
- **Determines types of metrics and failures and break down problems along the dependency path (how)**

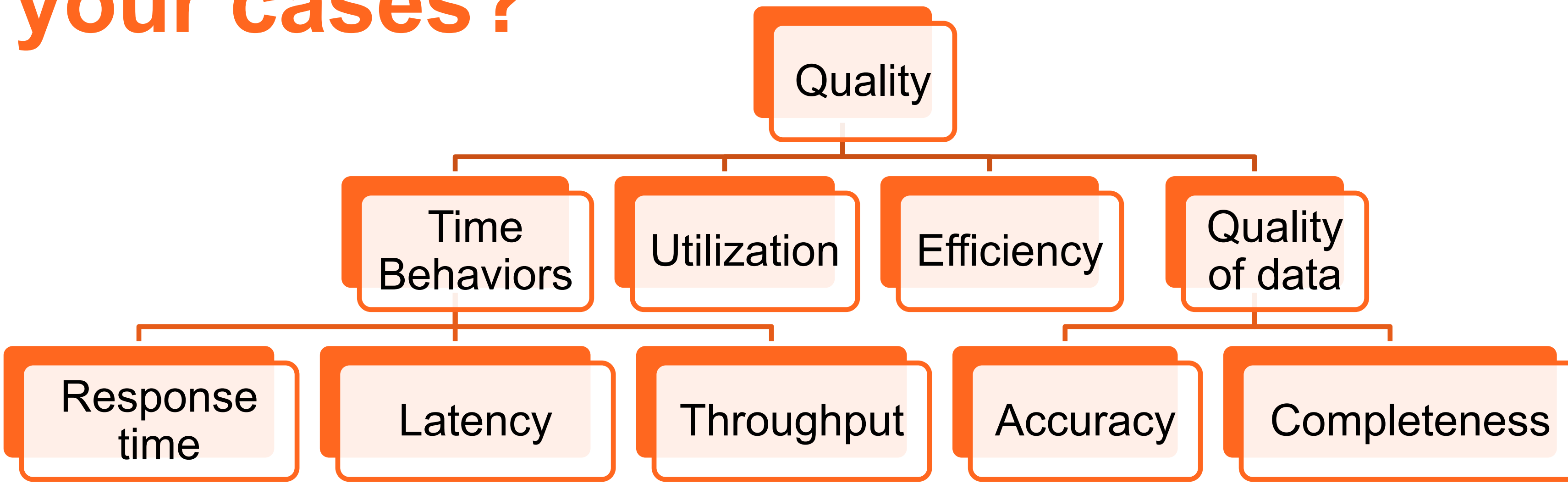
Boundaries and dependencies

Example of an ML serving with multiple models for object recognition (used in our hands-on)



- **Subjects for testing/debugging**
  - Data?
  - Model?
  - Underlying service platform?
  - Or all of them?

What are the most critical metrics for your cases?



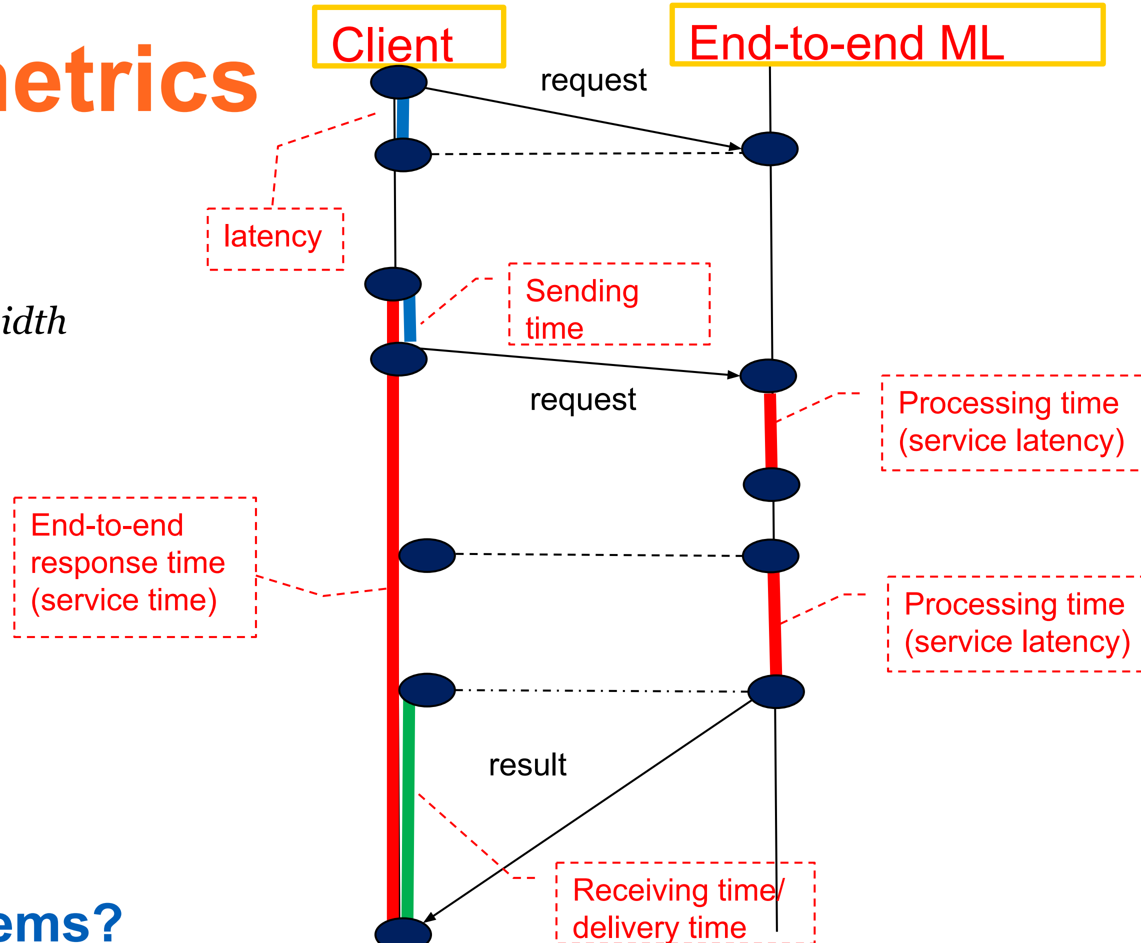
Industry view: https://guidingmetrics.com/content/cloud-services-industrys-10-most-critical-metrics/
NIST: https://www.nist.gov/sites/default/files/documents/itl/cloud/RATAX-CloudServiceMetricsDescription-DRAFT-20141111.pdf

Contradiction/Tradeoffs between Efficiency versus Resilience
Metrics for an ML model =! Metrics for ML system

Common performance metrics

- **Timing behaviors**
  - Communication
    - *Latency/Transfer time*
    - *Data transfer rate, bandwidth*
  - Processing
    - *Response time (service latency/time)*
    - *Throughput*
- **Utilization**
  - Network utilization
  - CPU utilization
  - Service utilization
- **Efficiency/Scalability**
  - Concurrent executions

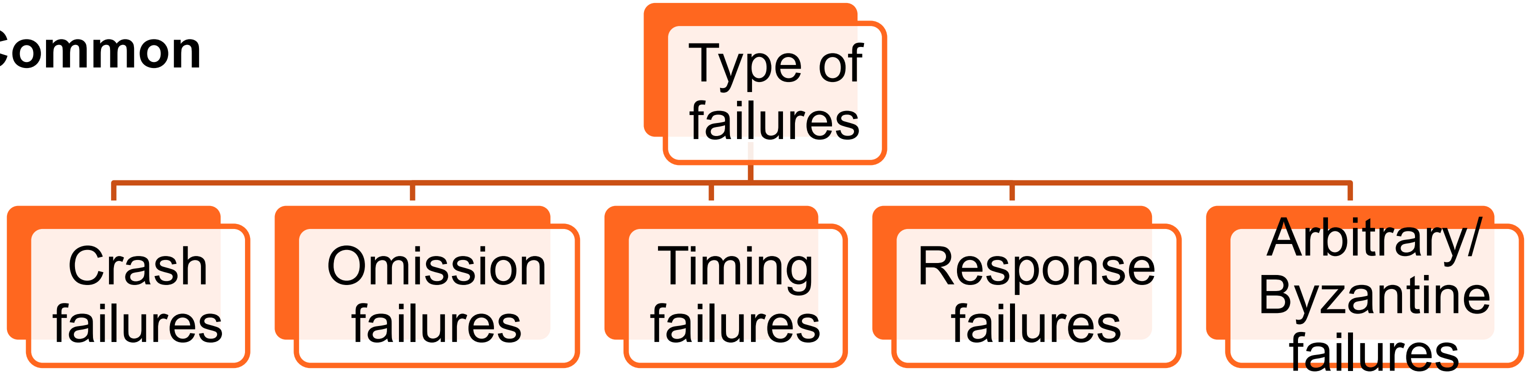
Examples



are they enough for ML systems?

Types of Failure

Common



AI: may do something that it should not do, because of wrong design or emergent/unintended behaviors

But unforeseen failures cannot be determined in advance ⇒ design for handling failures

Check: https://arxiv.org/pdf/1910.11015.pdf for a "Taxonomy of Real Faults in Deep Learning Systems"

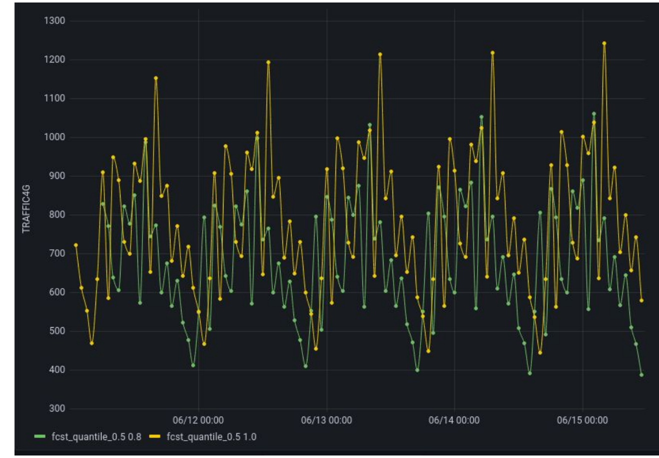
Metrics for Data

- Completeness
- Timeliness
- Currency
- Validity
- Format
- Accuracy
- Data Drift

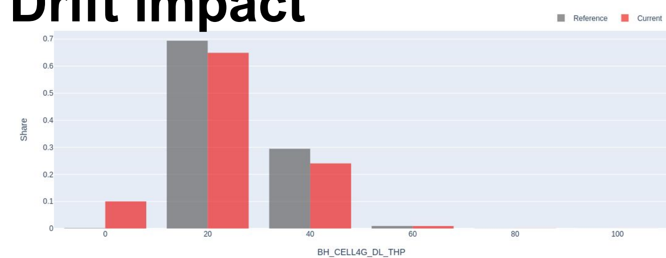
Understand the impact

Often evaluation methods are different for different types of data, metrics and when to evaluate

Forecasting



Drift impact



(examples with real mobile data)

Metrics for ML models

- **Confusion matrix**
- **Accuracy**
- **Loss**
- **True positive rate**
- **False positive rate**
- **F1 Score/F-measure**
- **Etc.**

(see https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234)

How would we define “reliable function” of the model? E.g., when should we “retrain” the model?

But how can we relate them to ML service/system?

Remember: each type of ML algorithms has different metrics

Explainability

- **Quality for performance optimization and service level agreements**
  - based on quality, we optimize a system to reduce operation costs, contract violations and improve customer satisfaction
- **Quality for explainability: for both users and providers**
  - explain the flows
    - *e.g. to where the data is sent and where is the inference service*
  - explain the service results
    - *e.g., what is the accuracy? is the result bias*
  - explain the cost
    - *e.g., why is the cost so high? which are cost components?*
- **Optimization and Explainability goals can be conflicted**

Aalto University
School of Science

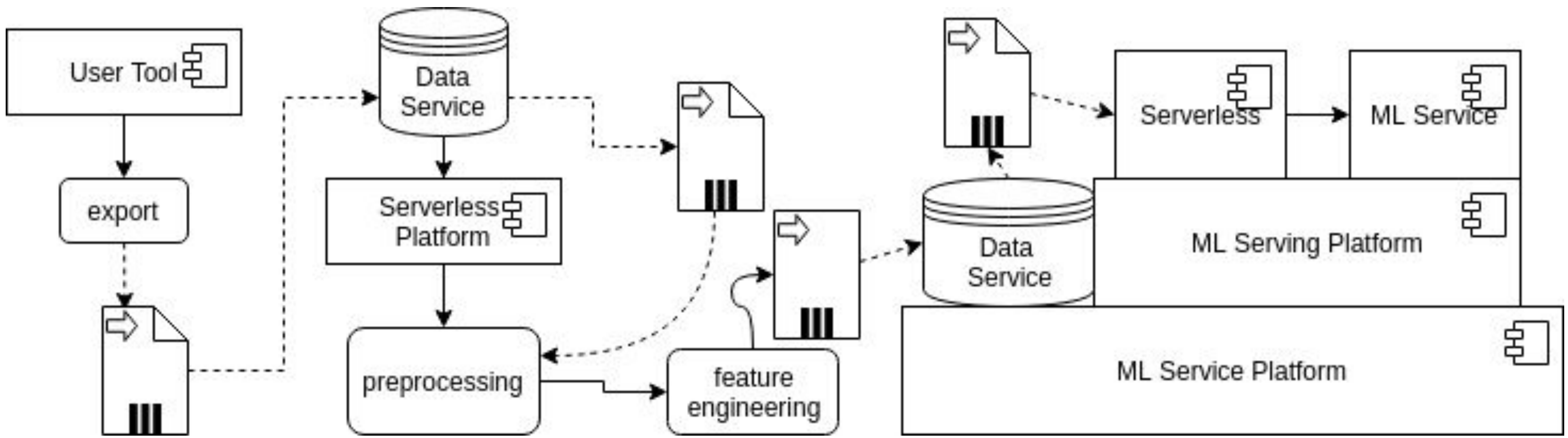
Benchmarking, Monitoring and Observability

Benchmarking

- **Benchmarking**
  - for comparing big data/ML systems w.r.t. selected (standard/common) workloads
- **Where to be benchmarked in an end-to-end system**
  - benchmark individual subsystems: message brokers and data ingestion, databases and ingestion/query, data processing, ML models, serving platform
- **What to be benchmarked**
  - Metrics:
    - *data ingestion throughput, processing throughput and time, component CPU and memory, training and inferencing time and accuracy*
  - Scenarios/Use cases: autonomous vehicles, vision, time series forecast, etc.

Benchmarking

If you have an end-to-end ML system, does it make sense to benchmark the whole system?



Benchmarking - ML

Examples:

ML Commons

Division/Power	Round	Organization1	System Name	Accelerator	Processor	Software1	MLC Model	# of Accelerators	Availability1	Scenario1
Closed	v4.1	(All)	(All)	(All)	(All)	(All)	(All)	(All)	(All)	(All)

MLPerf Results

MLCommons data as of: 8/28/2024 2:55:35 PM

Benchmark: **Inference**
System Type: **Edge**
Division/Power: **Closed**
Availability: **All**
Round: **v4.1**

						Benchmark1 / Model MLC1 / Scenario1 / Units1						
						net		retinanet			stable-diffusion-xl	
						Stream	Offline	MultiStream	SingleStream	Offline	SingleStream	Offline
Public ID1	Organization1	Availability1	System Name (click + for details)	Accelerator	# of Accelerators	y (ms)	Samples/s	Latency (ms)	Latency (ms)	Samples/s	Latency (ms)	Samples/s
4.1-0012	ConnectTechInc	available	NVIDIA Jetson AGX Orin 64G (TensorRT) + CTI For..	NVIDIA Jetson AGX Ori..	1							
4.1-0023	Dell	available	Dell PowerEdge XR8620t (1x L4 TensorRT)	NVIDIA L4	1			42.02	4.83	222.62		
4.1-0041	Lenovo	available	ThinkEdge SE455 V3 (2x NVIDIA L40S TensorRT)	NVIDIA L40S	2	0.33	86,304.60	10.39	1.81	1,629.08		
4.1-0042	Lenovo	available	ThinkEdge SE360 V2 (2x NVIDIA L4 TensorRT)	NVIDIA L4	2	0.39	25,600.00	32.19	3.81	453.67		
4.1-0051	NVIDIA	available	NVIDIA Jetson AGX Orin Developer Kit 64G (Tensor..	NVIDIA Jetson AGX Ori..	1						9,966.78	0.10
4.1-0067	UntetherAI	available	Supermicro SuperServer H13 (1x speedAI240 Slim)	UntetherAI speedAI240 ..	1	0.12	56,277.10					
4.1-0068	UntetherAI	available	Supermicro SuperServer H13 (3x speedAI240 Slim)	UntetherAI speedAI240 ..	3	0.12	168,720.00					
4.1-0069	UntetherAI	available	Dell PowerEdge R760xa (4x speedAI240 Slim)	UntetherAI speedAI240 ..	4	0.12	221,845.00					
4.1-0077	UntetherAI	preview	Supermicro SuperServer H13 (1x speedAI240 Previe..	UntetherAI speedAI240 ..	1	0.12	70,348.20					
4.1-0078	UntetherAI	preview	Supermicro SuperServer H13 (2x speedAI240 Previe..	UntetherAI speedAI240 ..	2	0.12	140,625.00					

Source: https://mlcommons.org/benchmarks/inference-edge/

Also check:

https://www.benchcouncil.org/benchmarks.html#edgeai

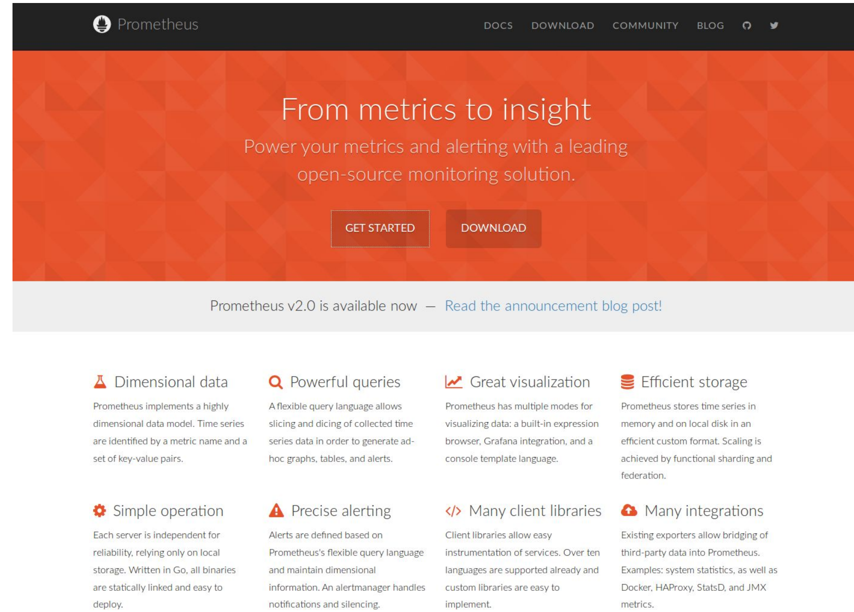
Service/Infrastructure monitoring tools

There are many powerful tools!

But only low-level, common well-identified monitoring data (infrastructures):

- pre-defined metrics exposed through interfaces with push/pull mechanisms

Distinguish between monitoring infrastructures/platforms vs monitoring data/metrics



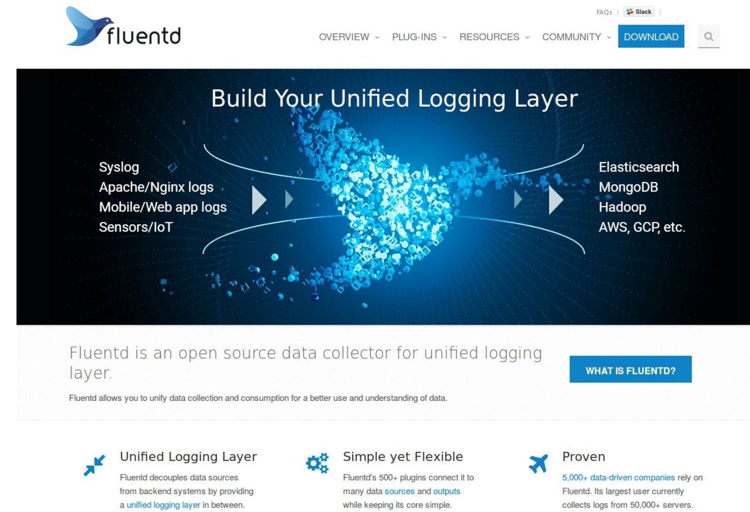
From: https://prometheus.io/

observIQ https://observiq.com/

Instrumentation for observability

Code instrumentation: for many metrics and logs that cannot be obtained from the outside of the component

the developer can instrument the code to capture metrics/generate logs/traces



From: https://www.fluentd.org/

Filebeat

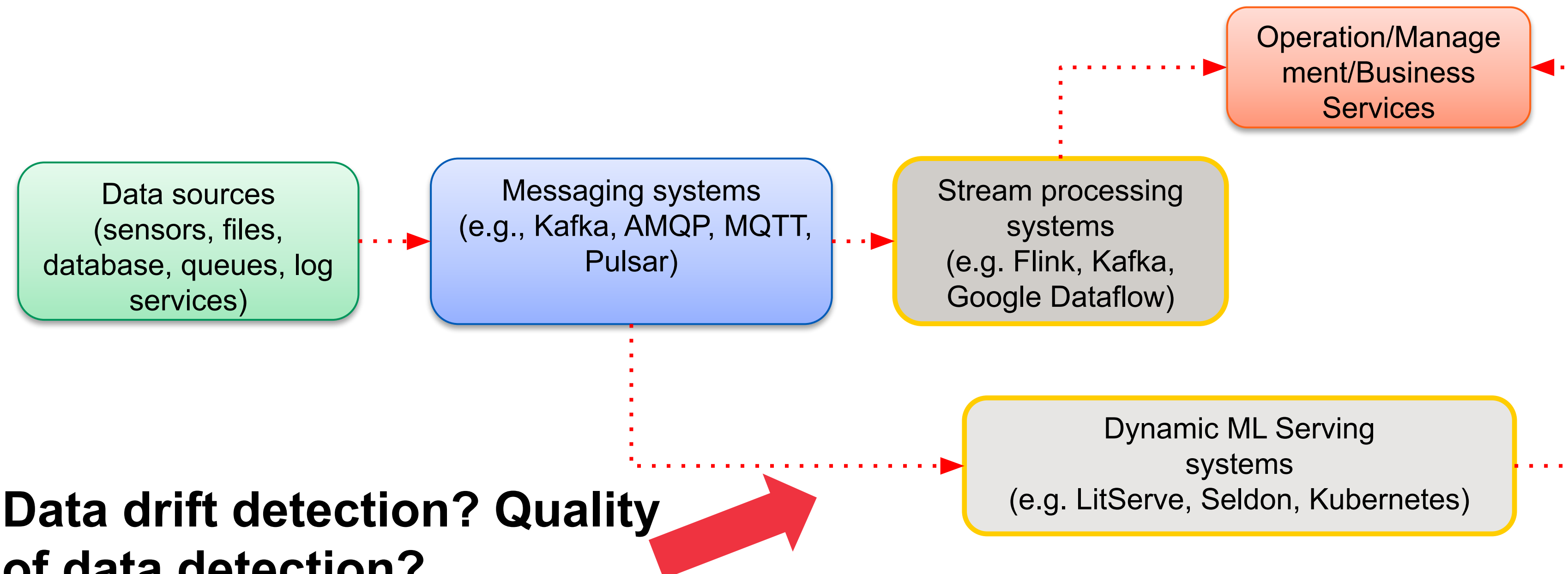
Lightweight shipper for logs

https://www.elastic.co/beats/filebeat

OpenTelemetry

https://opentelemetry.io/

Monitoring data metrics on-the-fly



Data drift detection? Quality of data detection?
Or performance prediction?

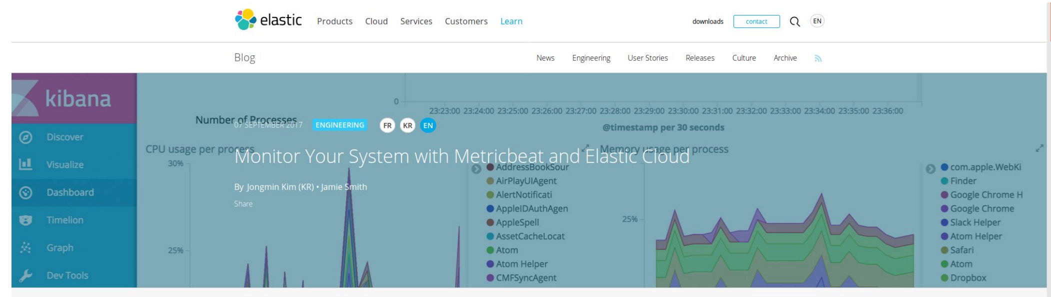
Tools for data quality

- **Generic tools/framework for checking data at rest**
  - Great expectation: https://github.com/great-expectations/great_expectations
  - YData (https://github.com/ydataai/ydata-quality)
  - Alibi-Detect (https://github.com/SeldonIO/alibi-detect)
  - Why-log (https://docs.whylabs.ai/docs/whylogs-overview/)
- **Integrated with processes in specific systems**
  - https://aws.amazon.com/blogs/industries/how-to-architect-data-quality-on-the-aws-cloud/
- **Working with specific data processing frameworks**
  - https://github.com/awslabs/python-deequ

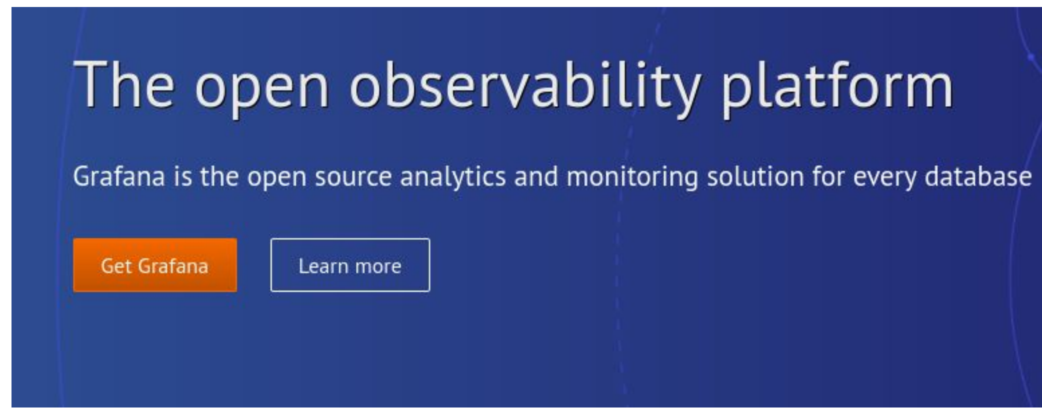
Visualization

Metrics and Visualization

- **Easy to visualize many types of metrics**
  - Human-in-the-loop
- **But only you can specify, define and map them to your structured applications**
- **Not for complex process automation!**
  - further integration and intelligence analytics (ML?)



https://www.elastic.co/products/kibana



https://grafana.com/

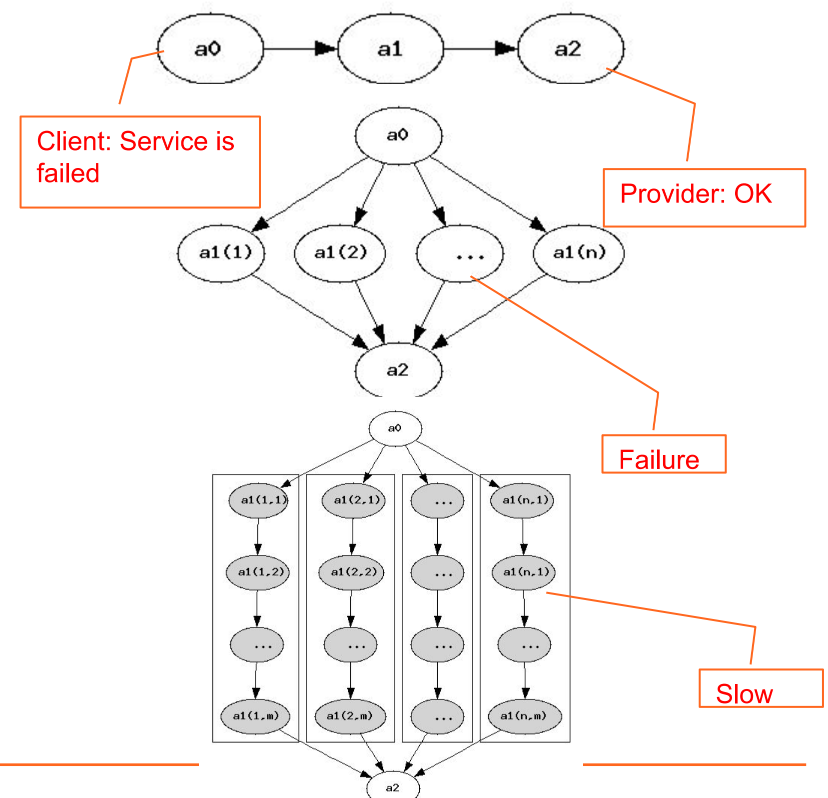
Observability

- **To monitor and understand the system as whole, end-to-end**
  - every component must be monitored
  - dependencies/interactions must be captured
  - diverse metrics, logs, tracing, etc. are needed to be integrated
- **Understand the states and behaviors of the whole systems**
- **Complex problems in big data/ML systems as these systems**
  - large-scale number of microservices in large-scale virtualized infrastructures
  - multi-dimensional states (code, models and data)

Understand the structure of big data/ML application

- **Composable method**
  - divide a complex structure into basic common structures
  - each basic structure has different ways to analyze specific failures/metrics
- **Interpretation based on context/view**
  - client view or service provider view?
  - conformity versus specific requirement assessment

Dependency Structure



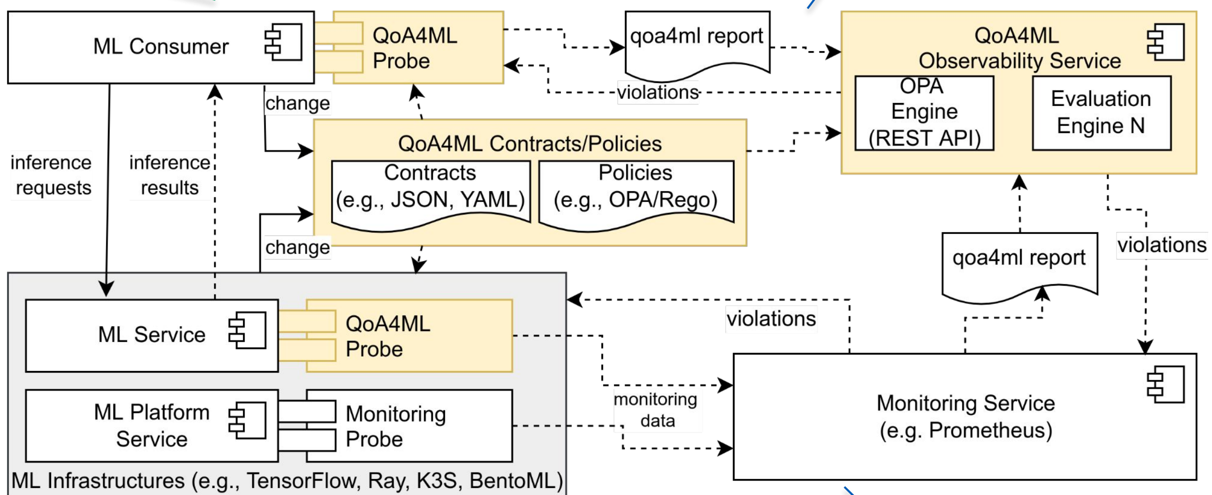
Support an end-to-end view or not

- **End-to-end reflects the entire system**
  - e.g., data reliability: from sensors to the final analytics/inference results
  - what if the developer/provider cannot support end-to-end?
- **The user expects end-to-end R3E**
  - e.g., specified in the expected accuracy
  - **Providers/operators want to guarantee end-to-end quality**
    - need to monitor different parts, each has subsystems/components
    - coordination-aware assurance, e.g., using elasticity

Example: ML contract observability with QoA4ML

Monitor data/ML-specific attributes

Design for different engines to be used



Reuse well-known monitoring systems

https://github.com/rdsea/QoA4ML

Experiment management

how do we manage important performance information for ML services?

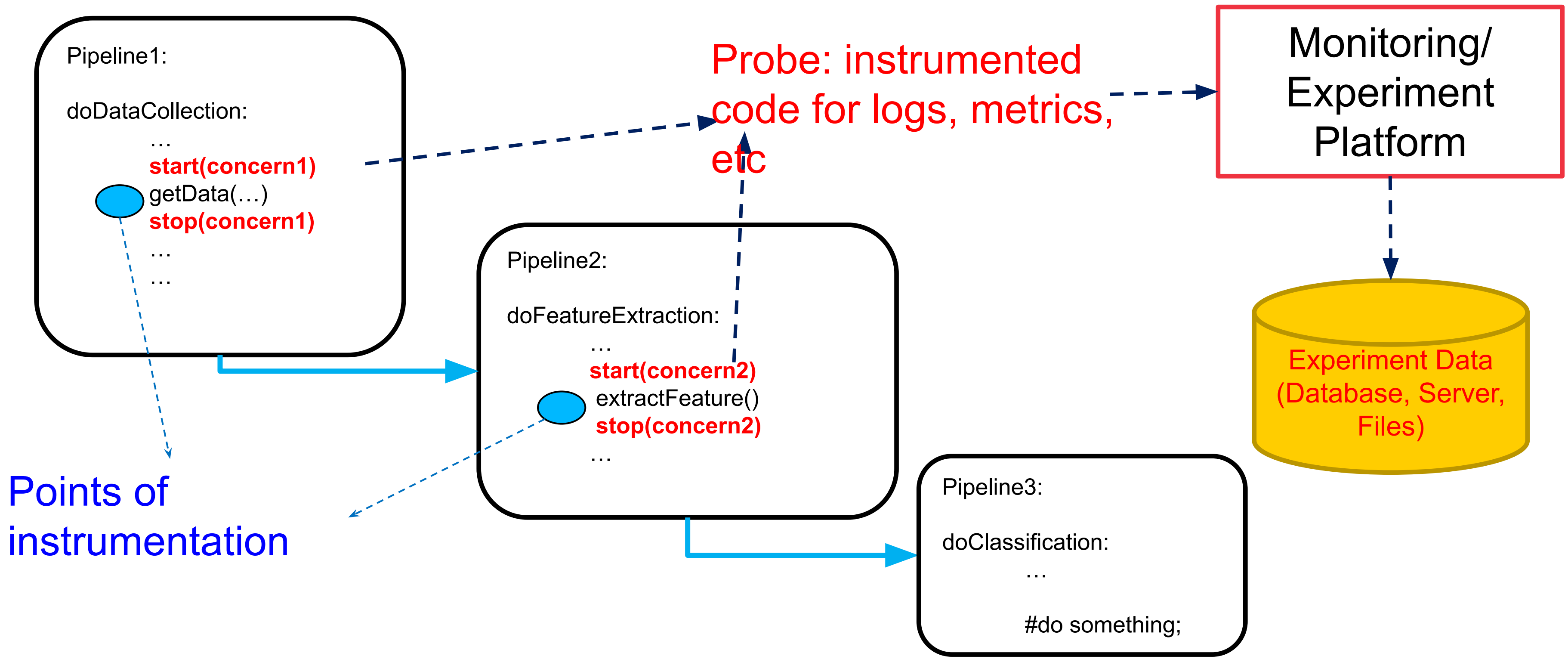
Problems

- **We need to run many experiments**
  - testability/observability purposes: figure out suitable configurations and parameters
  - how does this help to understand and support R3E?
- **Experiment management**
  - known domain and well-known books (e.g., "Design and Analysis of Experiments" by Douglas C. Montgomery)
  - principles: capturing various configurations and corresponding measurements given such configurations
  - how does it work for ML systems?
- **What do we need?**
  - tools/frameworks for tracking experiments

Notions

- **A single run/trial**
  - inputs, results, required software artefacts
  - computing resources, logs/metrics
- **Experiment**
  - a collection of runs/trials/executions gathered in a specific context
- **Steps**
  - parameterization: generate different parameters
  - deployment: prepare suitable environments
  - execution: run and collect metrics
  - analysis and sharing: analyze experiment data

Experiment tracking



But remember it is very large system! Different techniques/tools may be needed

Tools/Platforms

- **Tensorflow Board (https://www.tensorflow.org/tensorboard)**
- **Experiment in Azure ML SDK**
  - https://docs.microsoft.com/en-us/python/api/overview/azure/ml/?view=azure-ml-py#experiment
- **MLFlows**
  - https://mlflow.org/
- **DVC: https://dvc.org/**
- **Comet: https://www.comet.com/**
- **Weights and Biases:**
  - https://wandb.ai/site/experiment-tracking/

Examples: MLFlow APIs

- **Experiment**

  `mflow.start_run()/end_run()`

  `mflow.autolog()`
- **Logs/metrics collection**

  `mflow.set_tag()`

  `mflow.log_*()`
- **Tracking data management**
- **Local files, Databases, HTTP server, Databrick logs**

(follow our hands-on tutorial)

Experiment management: more than just ML models

- **Remember that there are many components in a system**
- **Experiment data about other components is also crucial**
  - have a full visibility and understanding of the system
  - support explainability and end-to-end optimization
- **ML model experiment must be combined with other types of experimental data**
  - experiment management for end-to-end systems
- **Edge LLMs and LLM applications/AI Agents in edge/cloud**
  - e.g., https://github.com/comet-ml/opik

Study log 2

Describe one big data/ML pipeline that you are familiar with and explain your thoughts on how would you support the aspects of "benchmarking", "monitoring", "observability", or "experimenting" for testing/implementing R3E aspects

Is enough to focus on 1 pipeline and 1 aspect

- **No "familiar pipeline" → look at our hands-on tutorials**
  - Be concrete, e.g., with metrics and possible tools
  - Analyze if things can be done easily or where are the challenges that might be interesting for further investigation
  - Optionally link to issues raised/addressed in a reading paper

Thanks!

Hong-Linh Truong
Department of Computer Science

rdsea.github.io