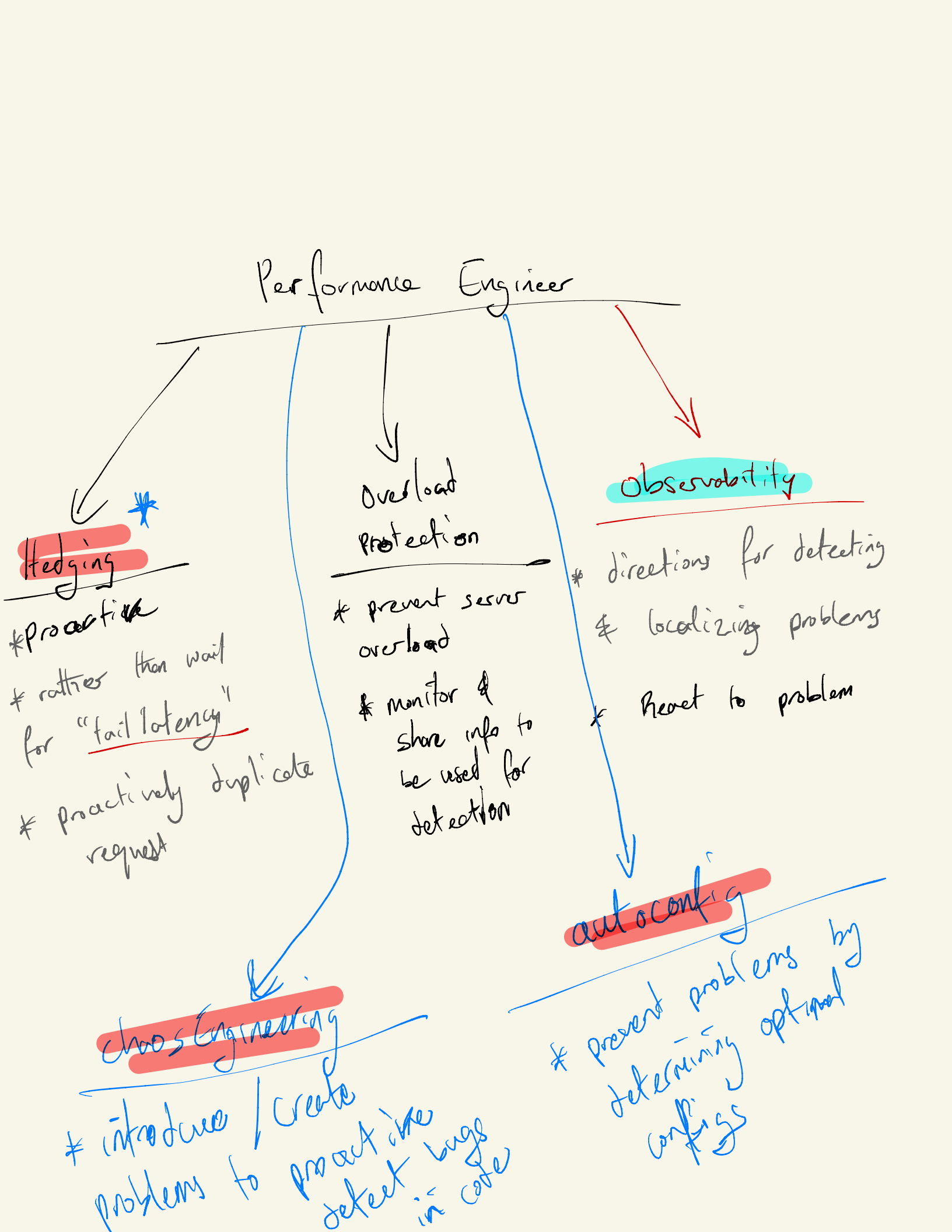

Performance Engineer
Hedging
*Proactive
* rather than wait for "fail latency"
* proactively duplicate request
Overload protection
* prevent server overload
* monitor & share info to be used for detection
Observability
* directions for detecting & localizing problems
* React to problem
autoconfig
* prevent problems by determining optimal configs
Chaos Engineering
* introduce / create problems to proactive detect bugs in code

"Three pillars of observability"

metrics (sexy)

timeseries data capturing resource use over time

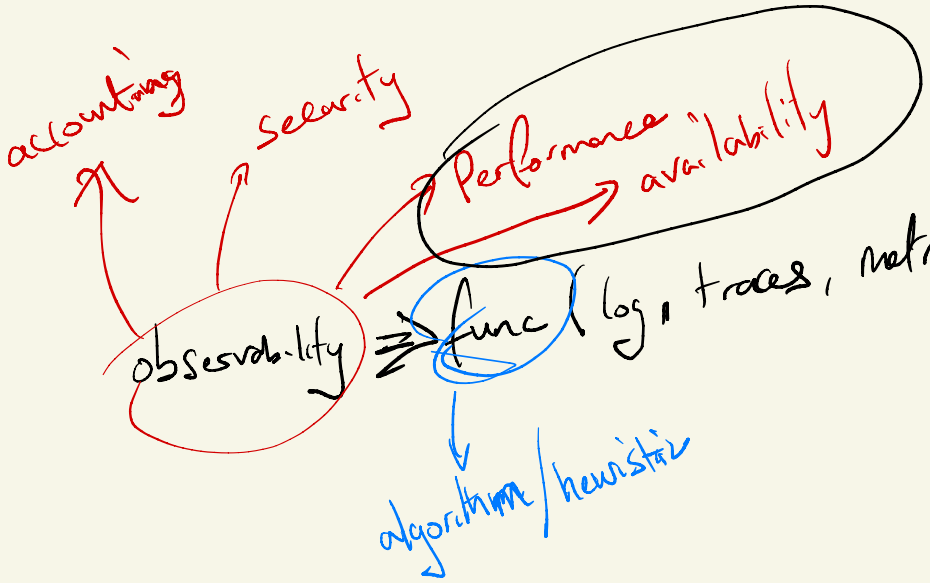
logs

recording of event in the system

distributed tracing

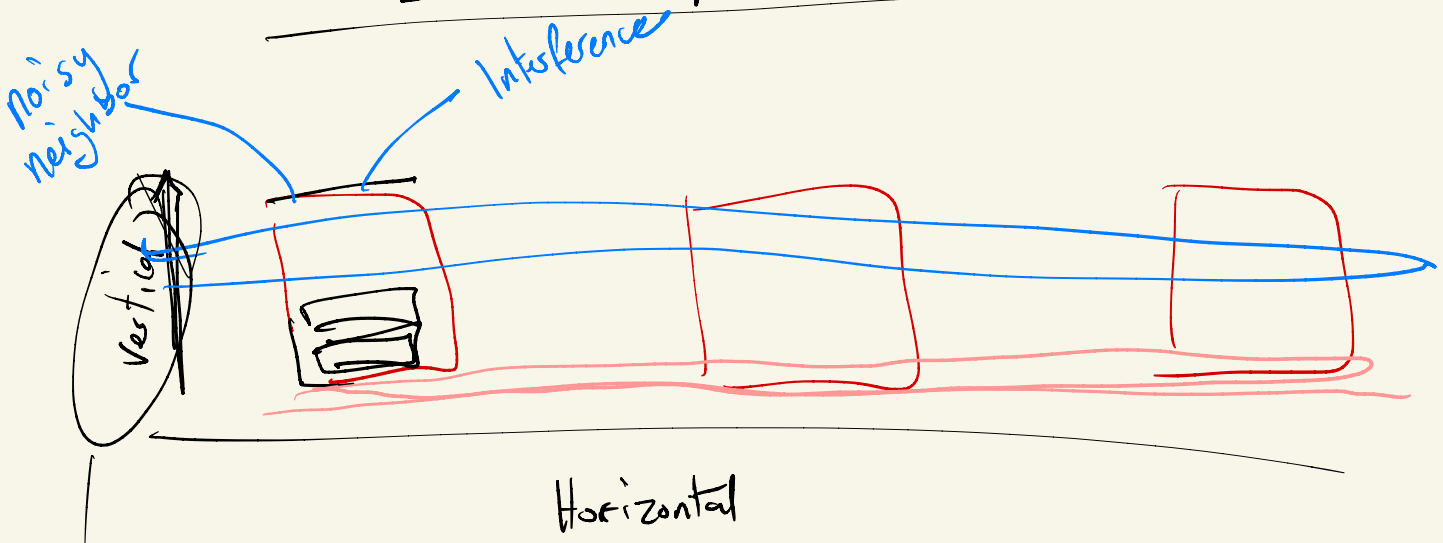
* Traces = follows the flow of execution across instance boundaries

captures instances & allows for localizing problem to instances



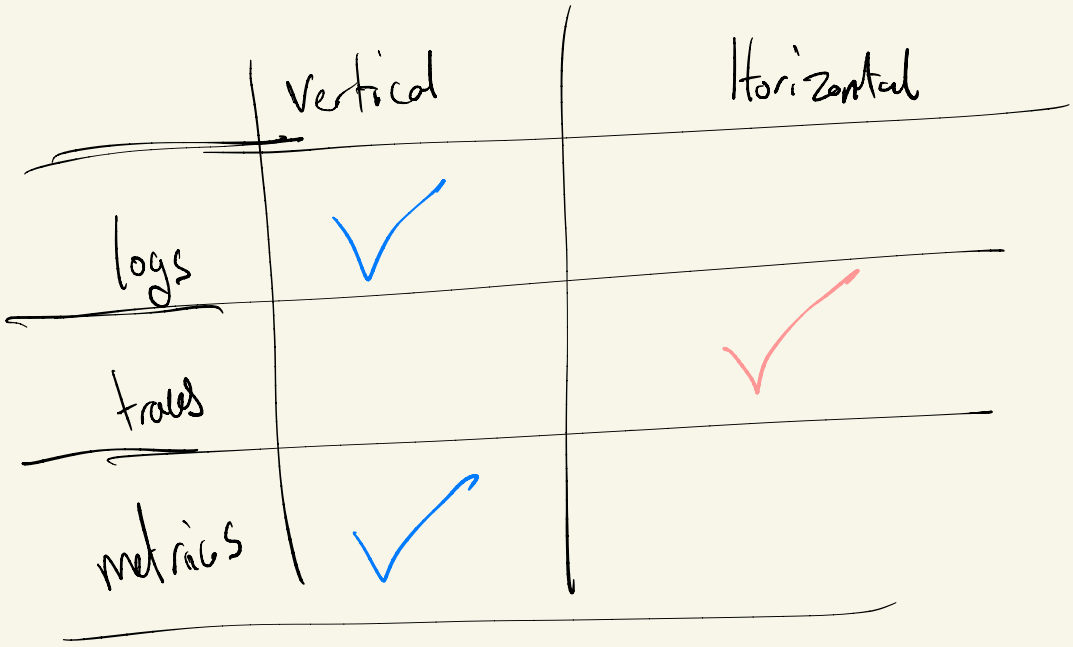
= internal workings of system

Dimensions of observality



interactions btw different entities on a machine

Noisy Neighbor : another service on the same node uses resources in a manner that impacts your ability to use resource hence bad perf

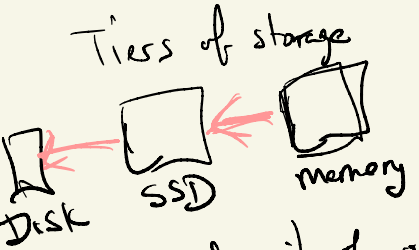


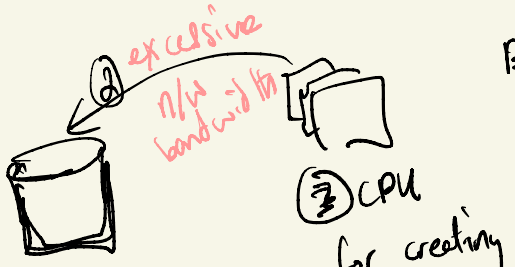
which events & which resources

Vertical ⇒ issues due to interactions on a machine

horizontal ⇒ issues due to interactions across machines

Data storage / sampling





Bottlenecks in sending data to a central DB

limited storage ①

(a) get rid of old or irrelevant data

(b) Sample data as you collect

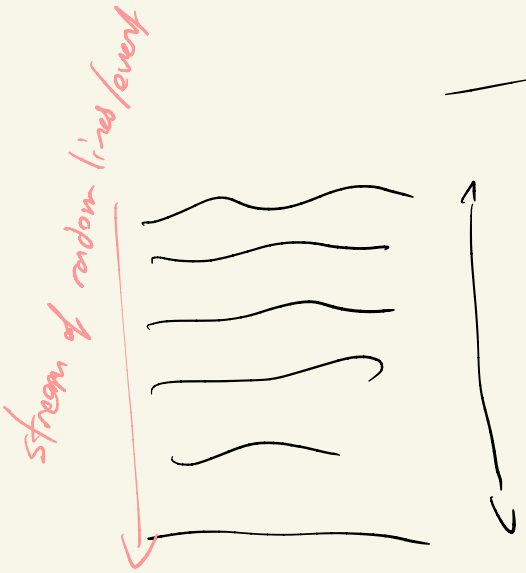
(c) Turn off observability data collection

Heterogeneity in services leads to variety in the quality of captured data

different sampling

to no data

Logs



fancy print statement

print("log data about CPU: %u @ time: %u")

① what are fundamental templates

② as a stream of templates

identify unusual templates

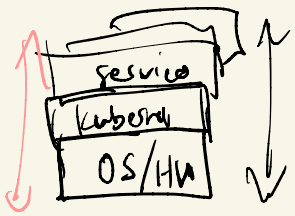
identify anomalous spike in relative # of templates

Trace / microservice based errors

Metrics V. Logs

Sieve => reduce overheads or speed up processing

unstructured
microservice / trace level



explosion in metrics

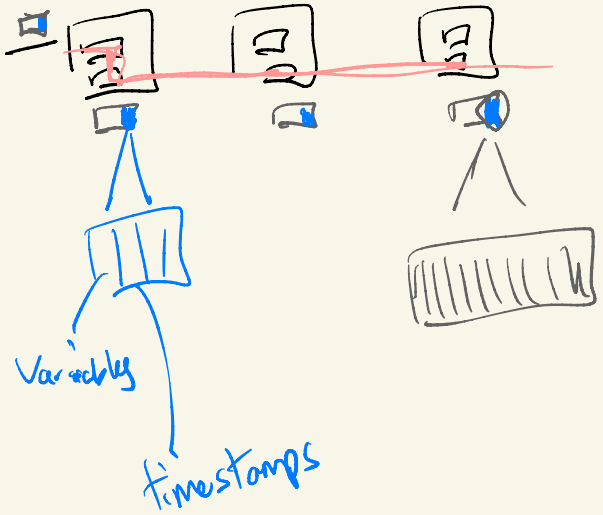
k-shape

reduce # of metrics collected / analyze

call-graph to help causality

restrict the pair of variables / metrics that granger causality

Trace



each instance/process involved in servicing a request

or each function that's required to process a request