Cloud Control Systems - Real-Time Analytics

Dr. Simon Tuffs
Cloudstream

http://tinyurl.com/qfyuyn2

Background

- University of Oxford
  - Self Tuning Control Systems/GPC
- 25 years software industry: highlights
  - Iridium Ground Station (Motorola)
  - Spacestation Infrastructure (Boeing)
  - Cloud (Netflix)
- Theme: Software at Scale

This Talk

- Cloud Based Service Applications
  - Analytics & Control, how and why
- Cloud Applications As:
  - Multi-variable time-series systems
  - Amenable to signal processing
  - Resilient to failure using analytics
  - Operational using feedback control

Cloud Application Architectures

Cloud: Application Architectures

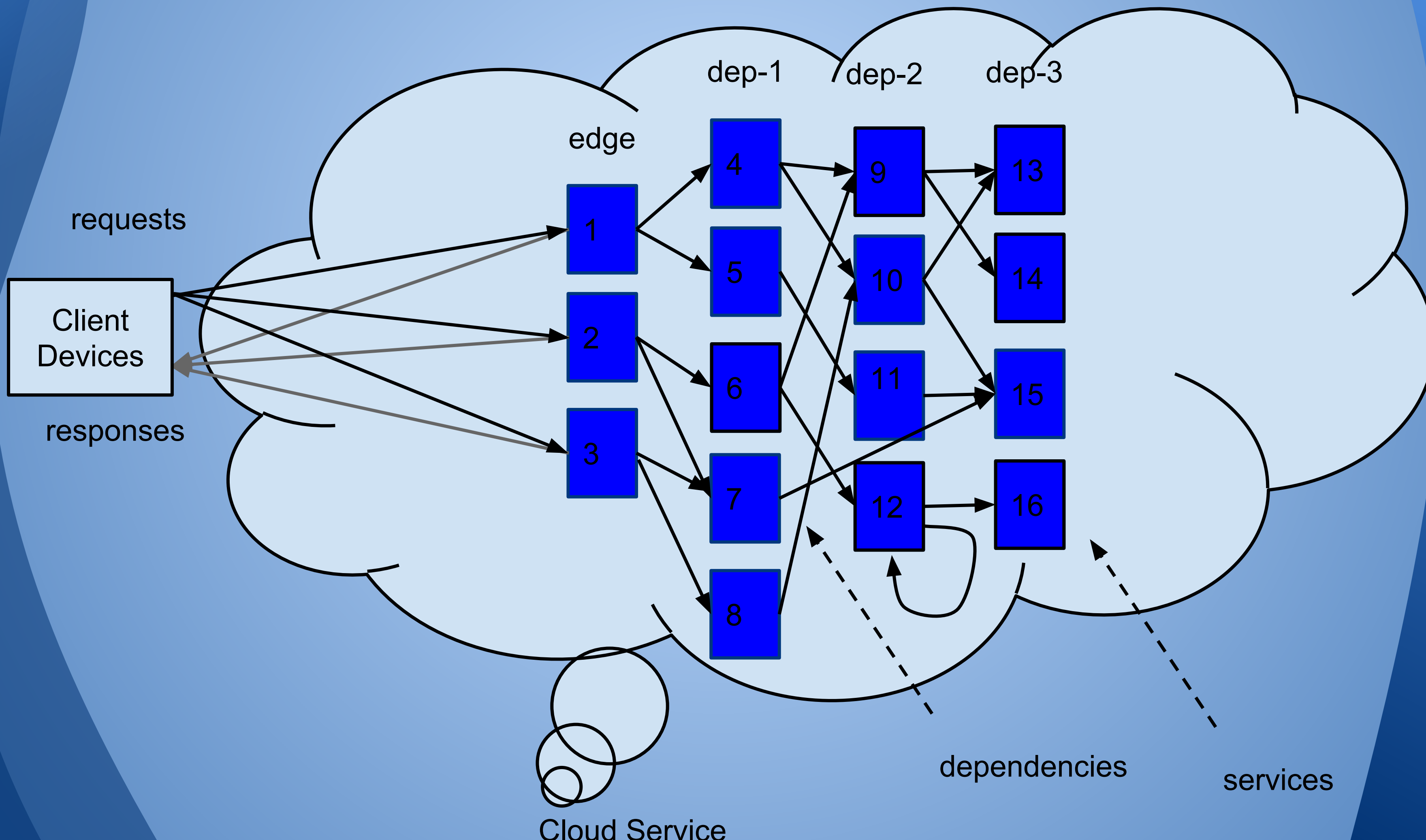
- Classes
  - Micro-service
    - Massive volume business operations (e.g. Netflix)
  - Big-Data
    - Terabytes of data, captured from streams into persistent stores.
    - Sparse compute intensive operations

Cloud: Application Architectures

- Cloud Application composed of services
- Services have dependencies (graph)
- Requests flow from the edge down
- Reponses flow back to edge
- Latencies accumulate forwards
- Errors propagate backwards
- Services are developed independently

Cloud: Application Architectures

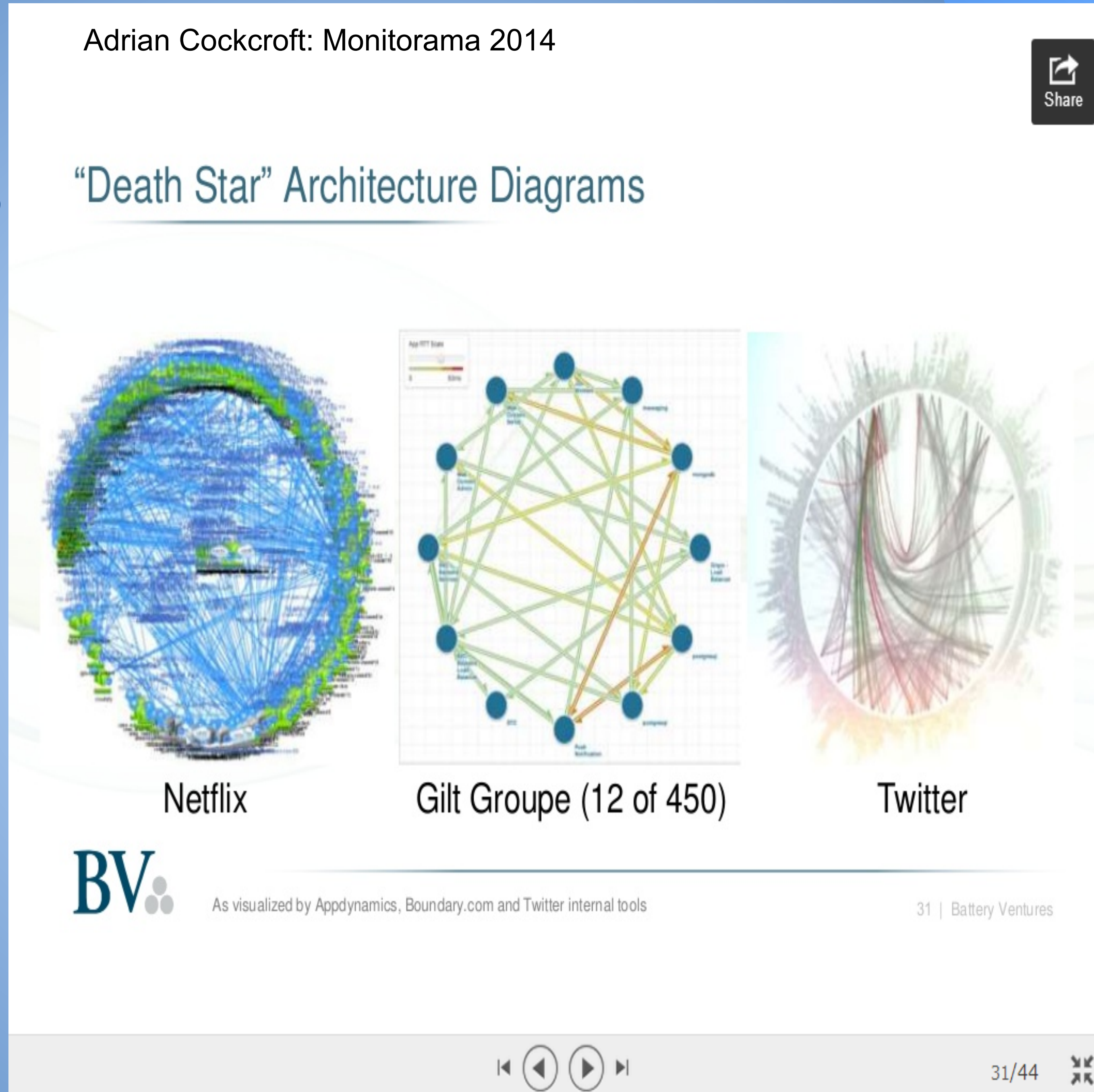
Services & Dependencies:



Cloud: Application Architectures

- But it isn't that simple
- A real dependency graph...
- "Death Star Architecture"

“We are not alone”



Cloud: Application Architectures

- Complex beyond human comprehension
- Nonlinear
- Time-varying
- Partially predictable
- Potentially chaotic
- The worst kind of “system”

Analytics are not optional, they are essential

Cloud Applications: Analytics Classes

- Operational
  - Availability, fault detection, repair, peformance optimization
- Business Intelligence
  - how much money are we making?
  - how many customers did we just lose?
  - how can we make more money?

Cloud Applications: Monitoring

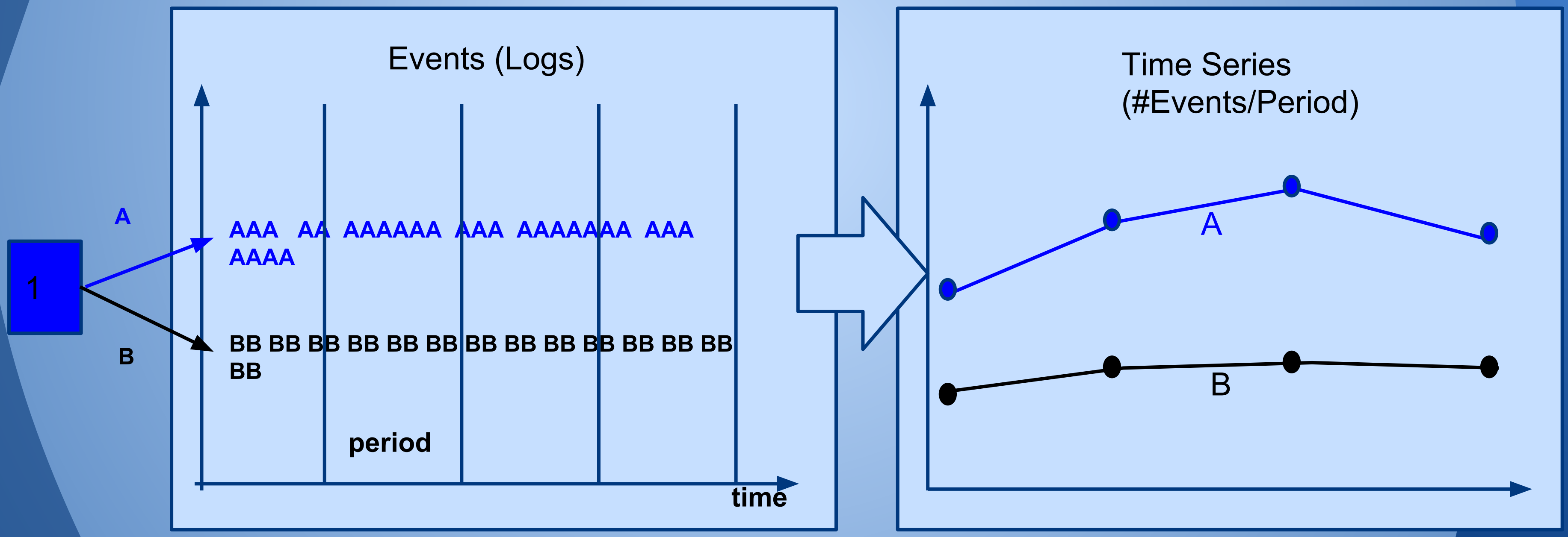
- To analyze you must monitor
- How do you handle billions of events?
- How do you transform them for analytics?

Cloud Applications: Monitoring

- Instrument services:
  - to expose internal details (e.g. type of errors, versus HTTP 503’s)
- With significant request volume:
  - monitored events become statistically driven time-series
  - signal processing methods then apply

Cloud: Monitoring

- From Events To Time Series:

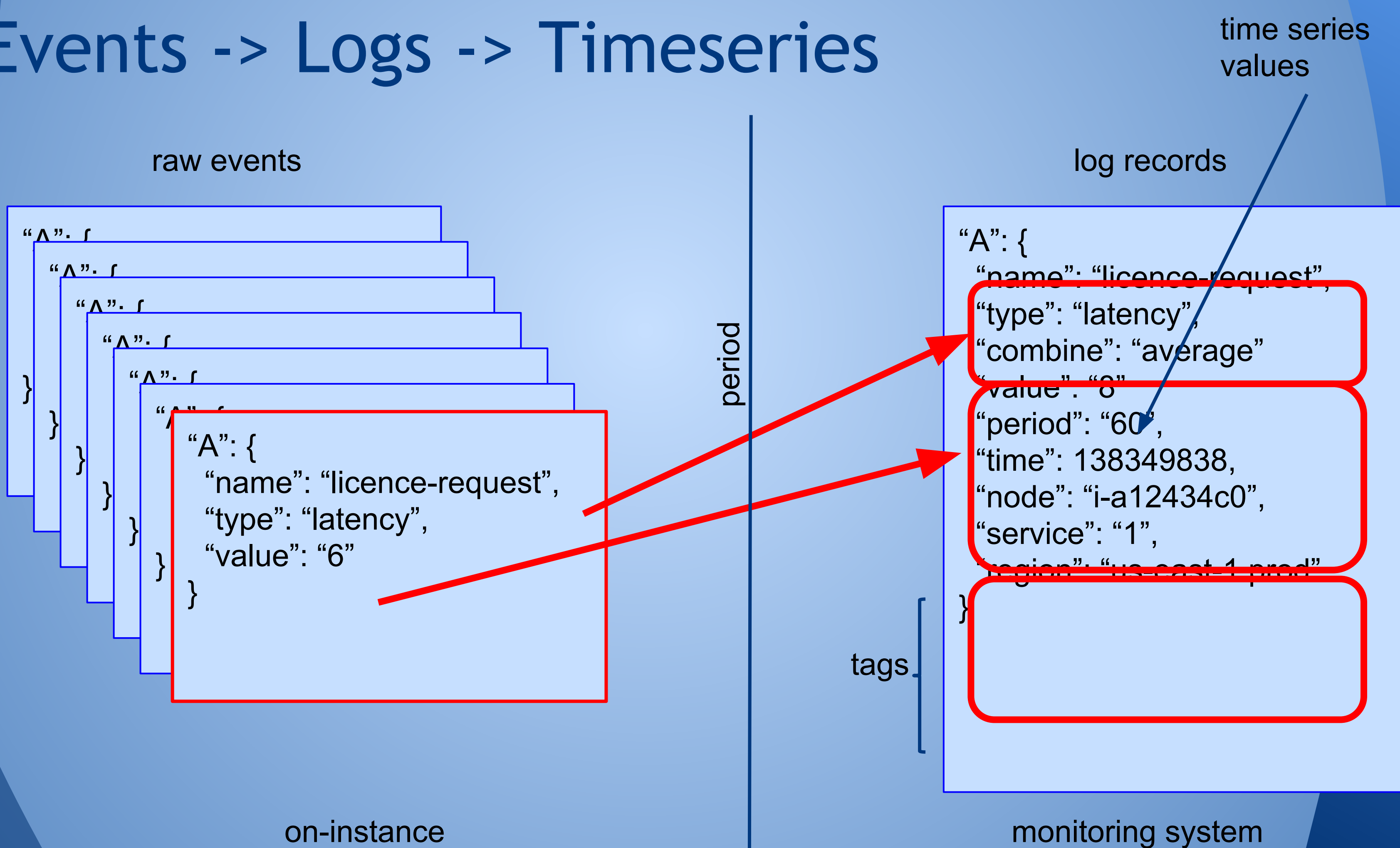


Cloud: Monitoring Architecture

- Convert events to time-series (coordinate transform)
  - bucket by period
  - classify & tag
  - store for query/retrieval
- Reduces dimension of data by many orders of magnitude
  - -> Real Time Analytics become feasible

Cloud: Monitoring Architecture

- Events -> Logs -> Timeseries



What to Monitor?

- “Assume that any metrics not being analyzed will turn out to be garbage”
  - Adrian Cockcroft, Architect Netflix Cloud
- Instrument to measure:
  - health (success, failure)
  - performance (load, cpu)
  - availability (timeouts, fallbacks)
  - resources (disk i/o, memory, handles),
  - sla’s (latency)

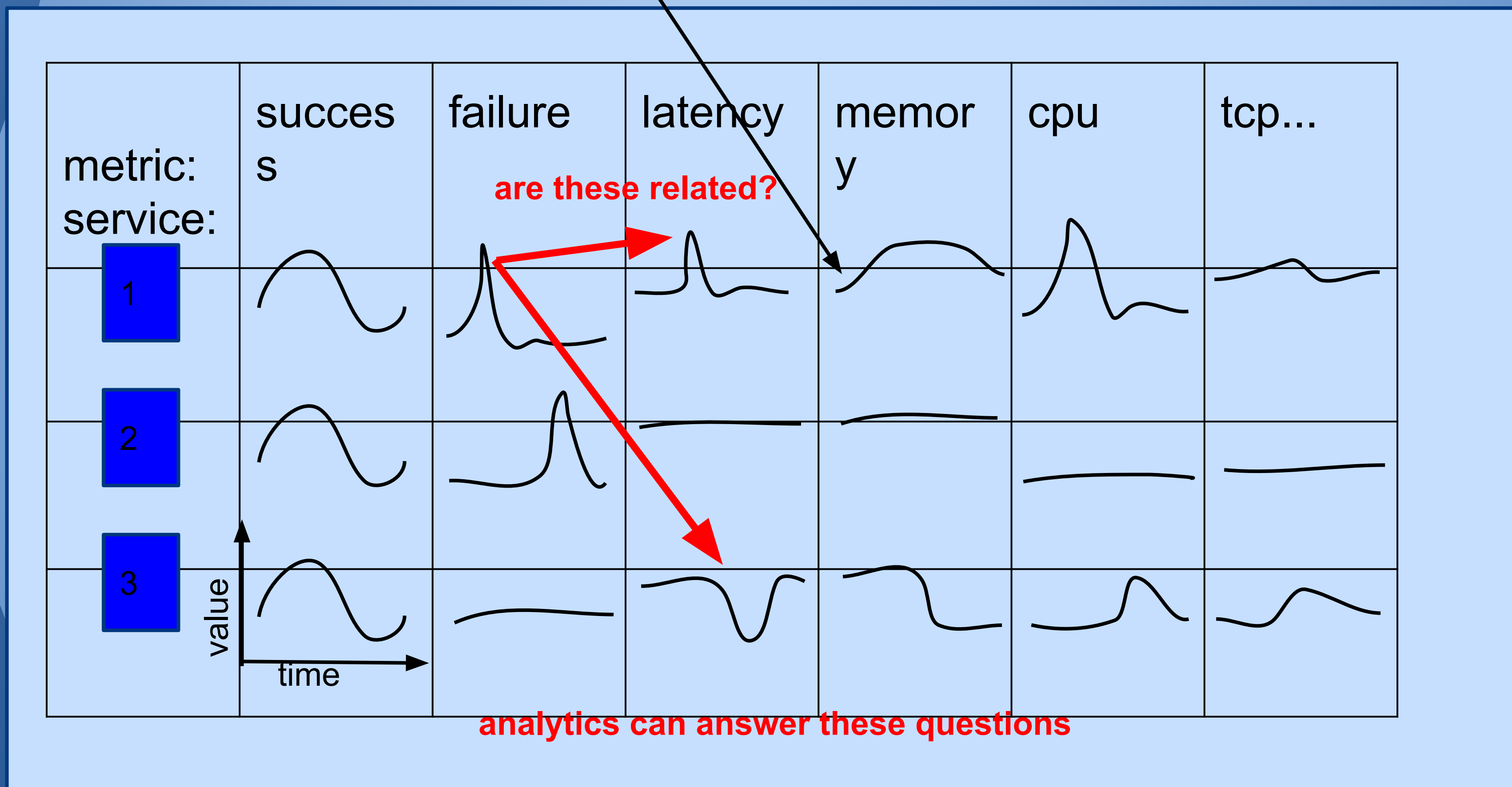
Visualization as an Analytic

Service Metric Visualization

- Classify metrics by type
- View services as rows of service:metrics
- Patterns start to emerge between visually.
- This scales to 100’s of services and metrics (make the graphs small, human visual cortex sees patterns)

Cloud: Visualizing

- service:metric



Beyond Visualization: Computational Analytics

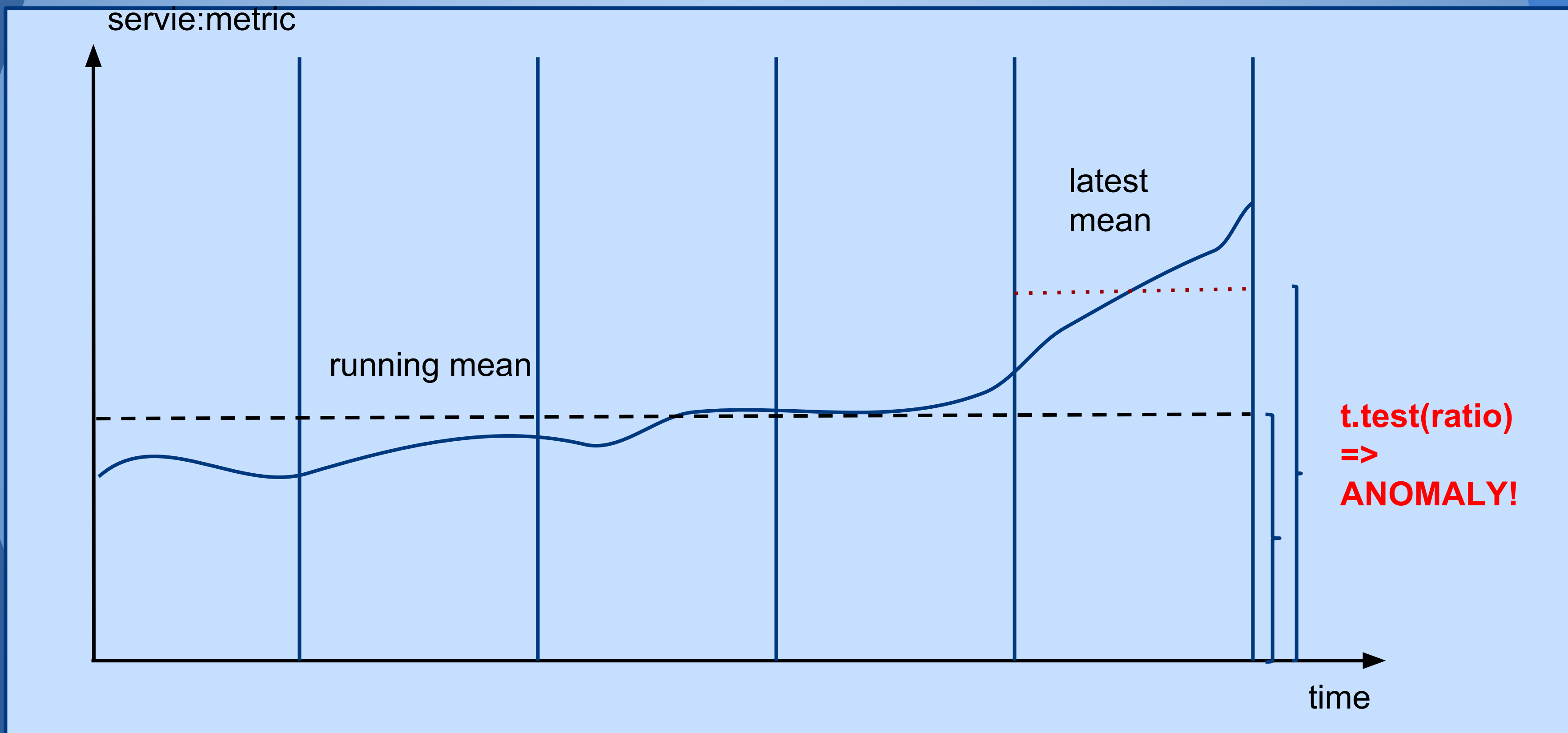
Anomaly Detection & Diagnosis

Anomaly Detection

- Look at a service:metric
- Is it behaving normally, or is it showing signs of distress?
- How can we automate this?
- Without lots of configuration?
- In a scale invariant way?
- Use a mean-shift analytic...

Analytics for Anomalies?

- mean? variance?

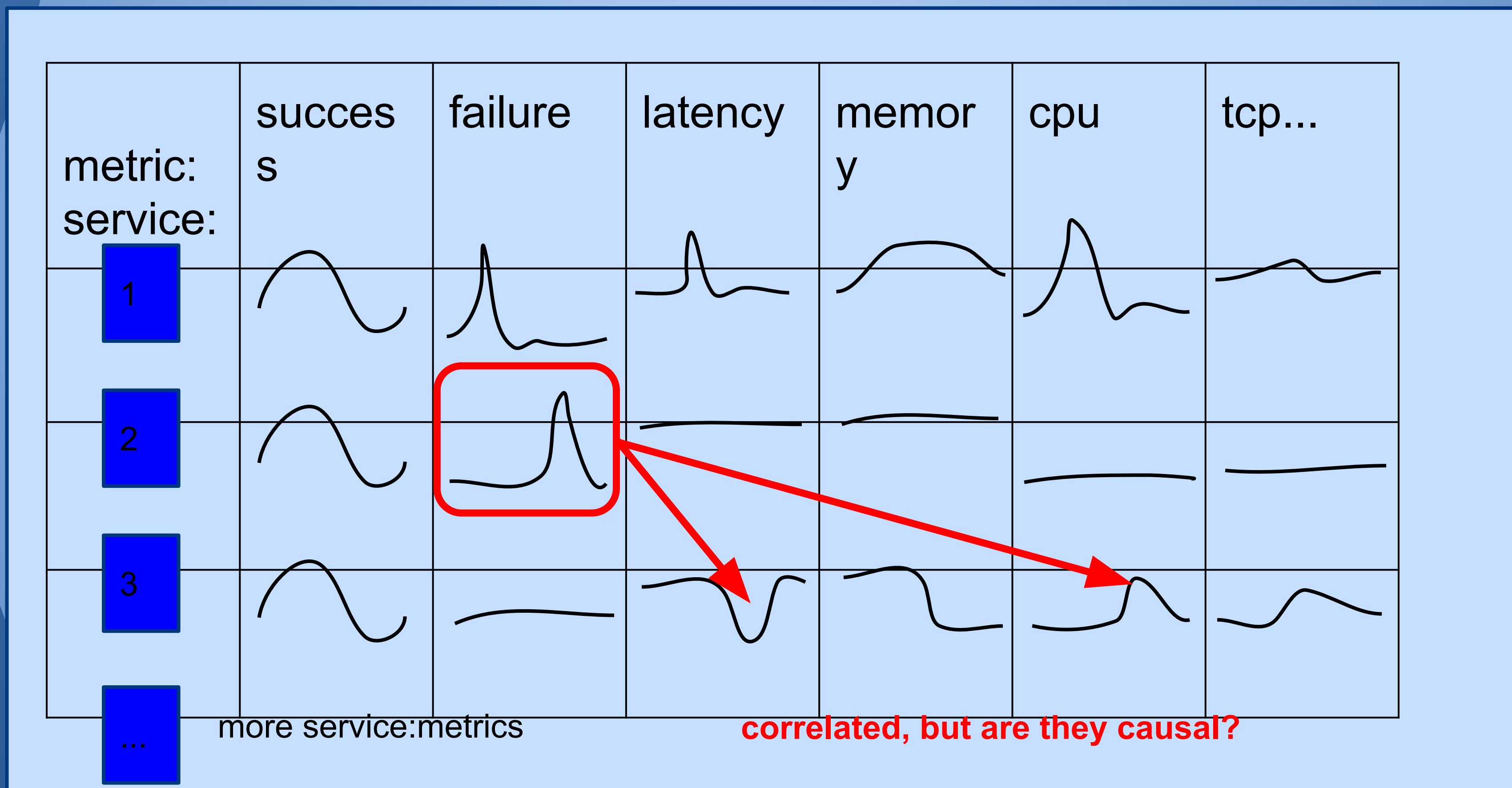


Analytics for Anomalies?

- You found an anomalous service:metric, now what?
- Correlate against *all other* service metrics
- This is fast (<0.1s for 400sm in R)

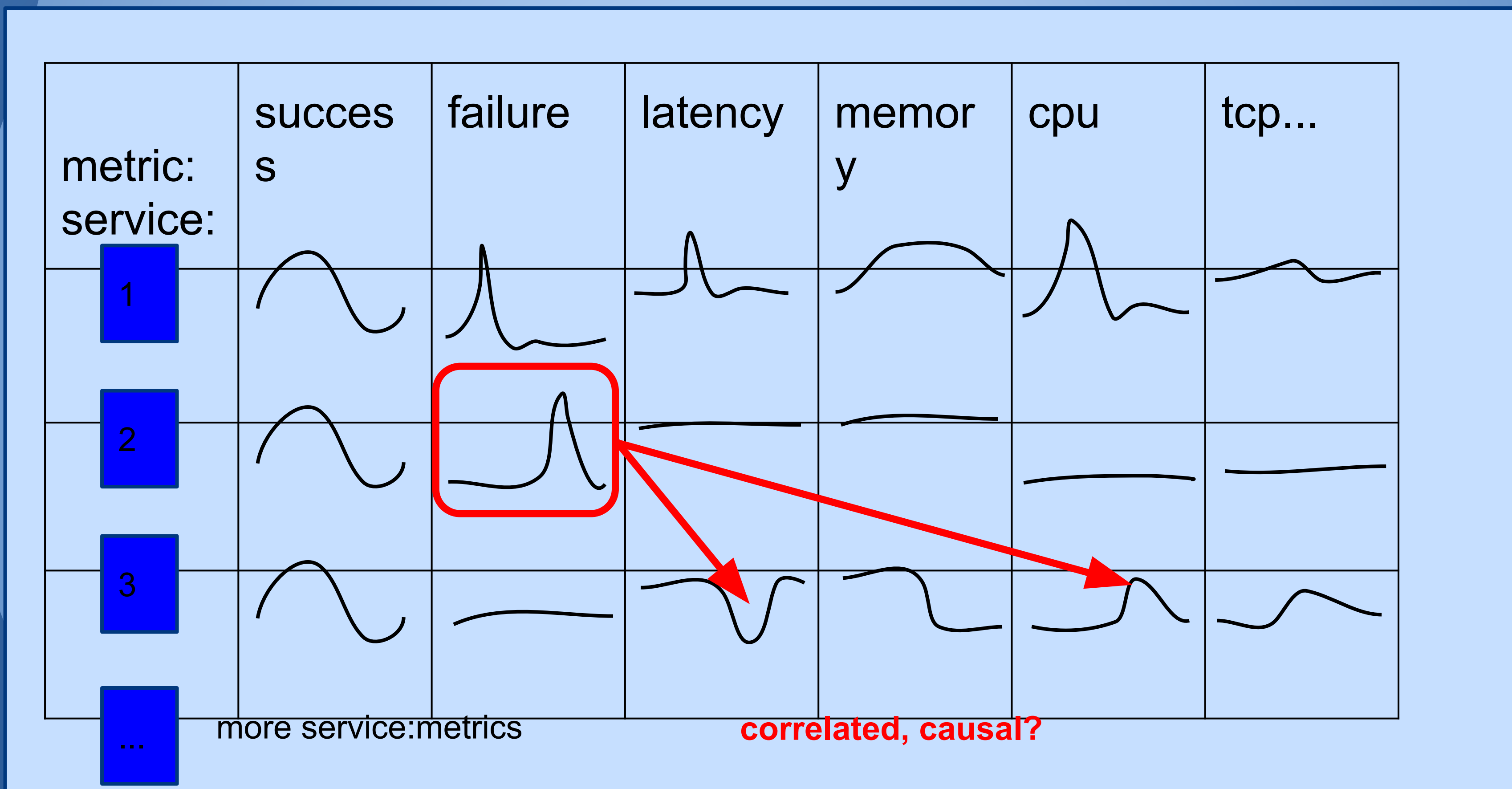
Correlate

- Pearson + mean removal



Filter

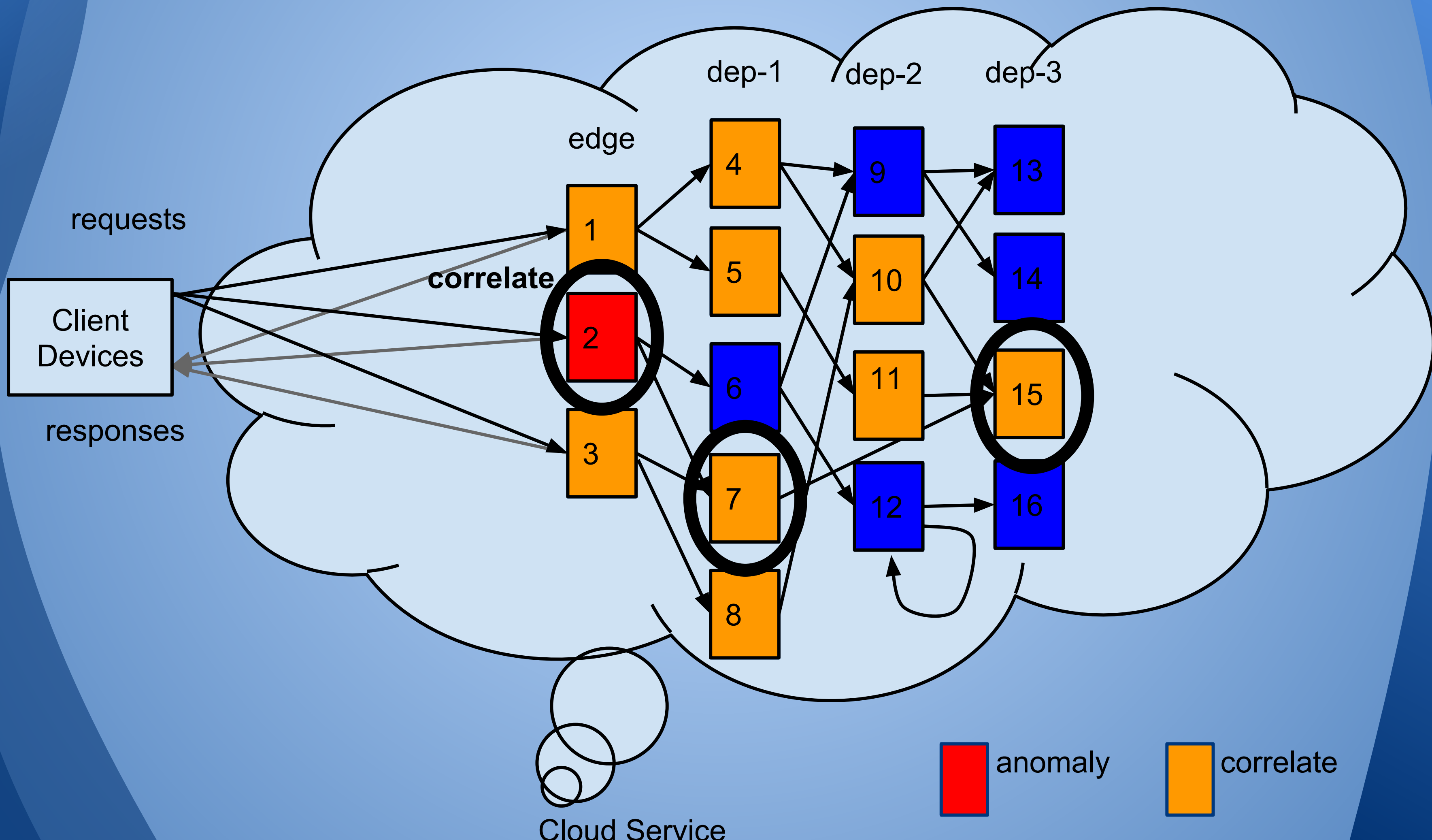
- Increase signal-to-noise:



Can we do more?

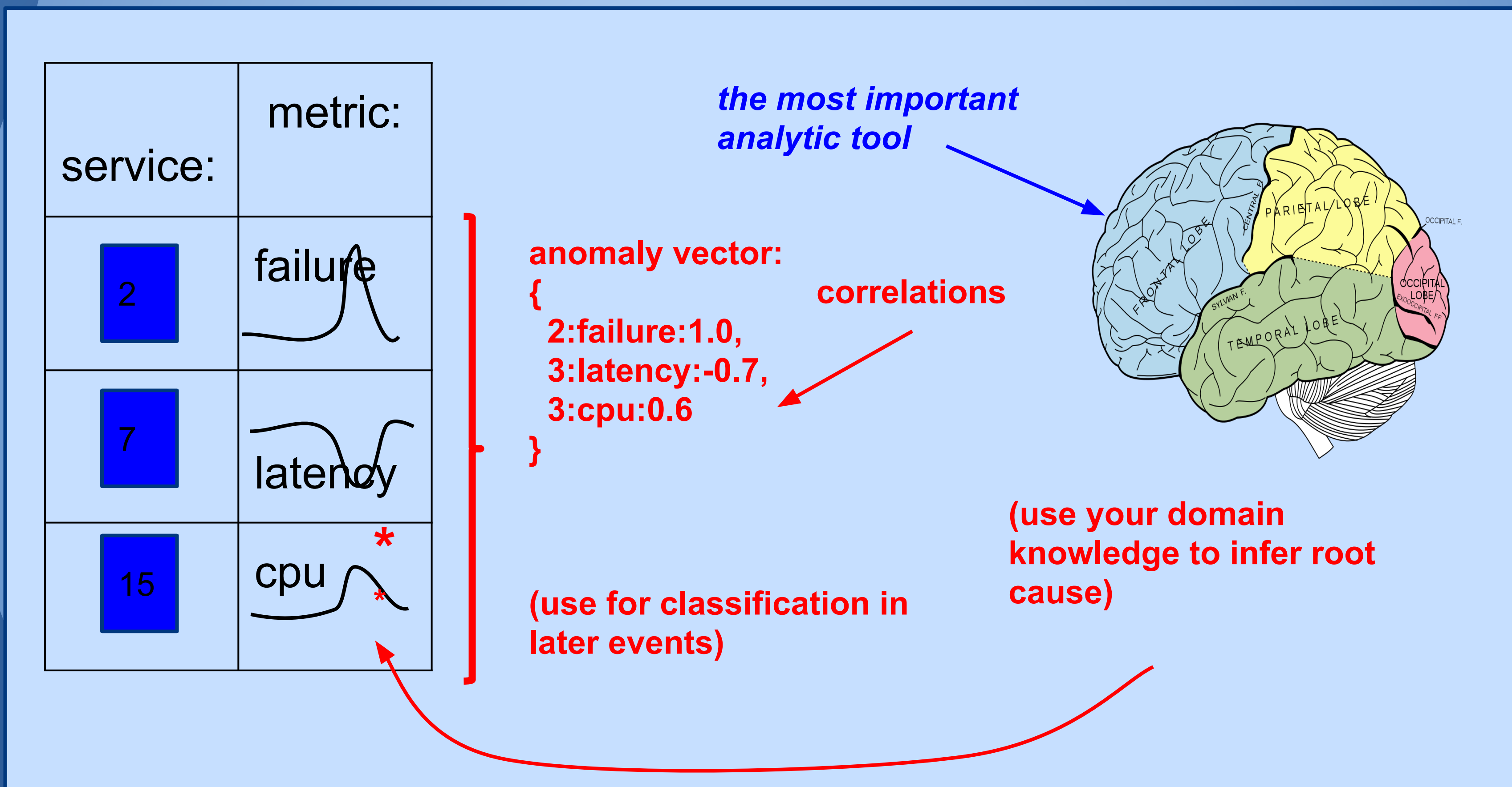
- Correlation x Dependency = Probable Cause

Anomaly -> Correlation -> Cause



Classify and Decide.

- Prune with dependency tree



Build a model

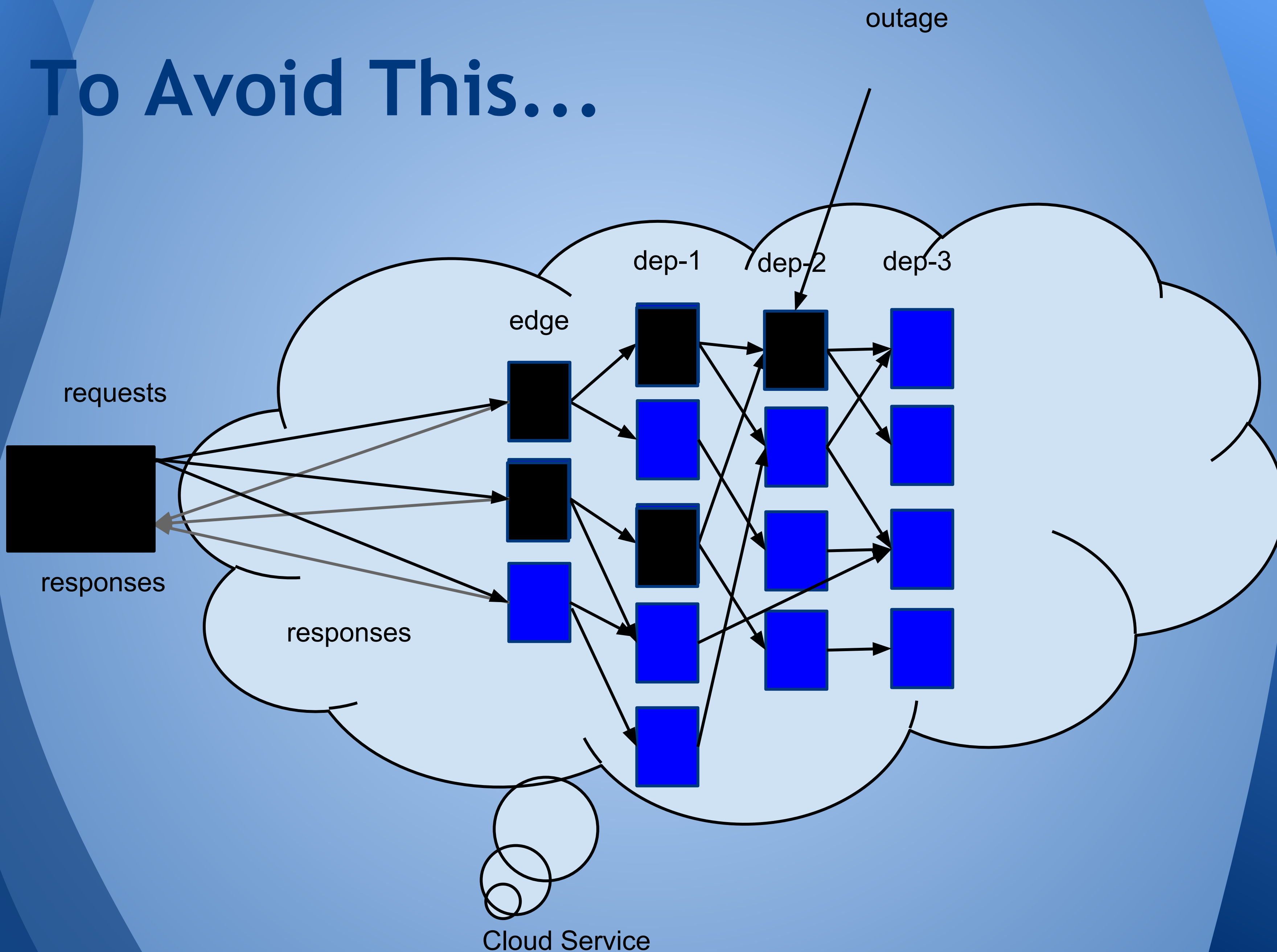
- Persist this pattern for future causal analysis
- Did we see this anomaly vector before?

Canary Analysis (deployment)

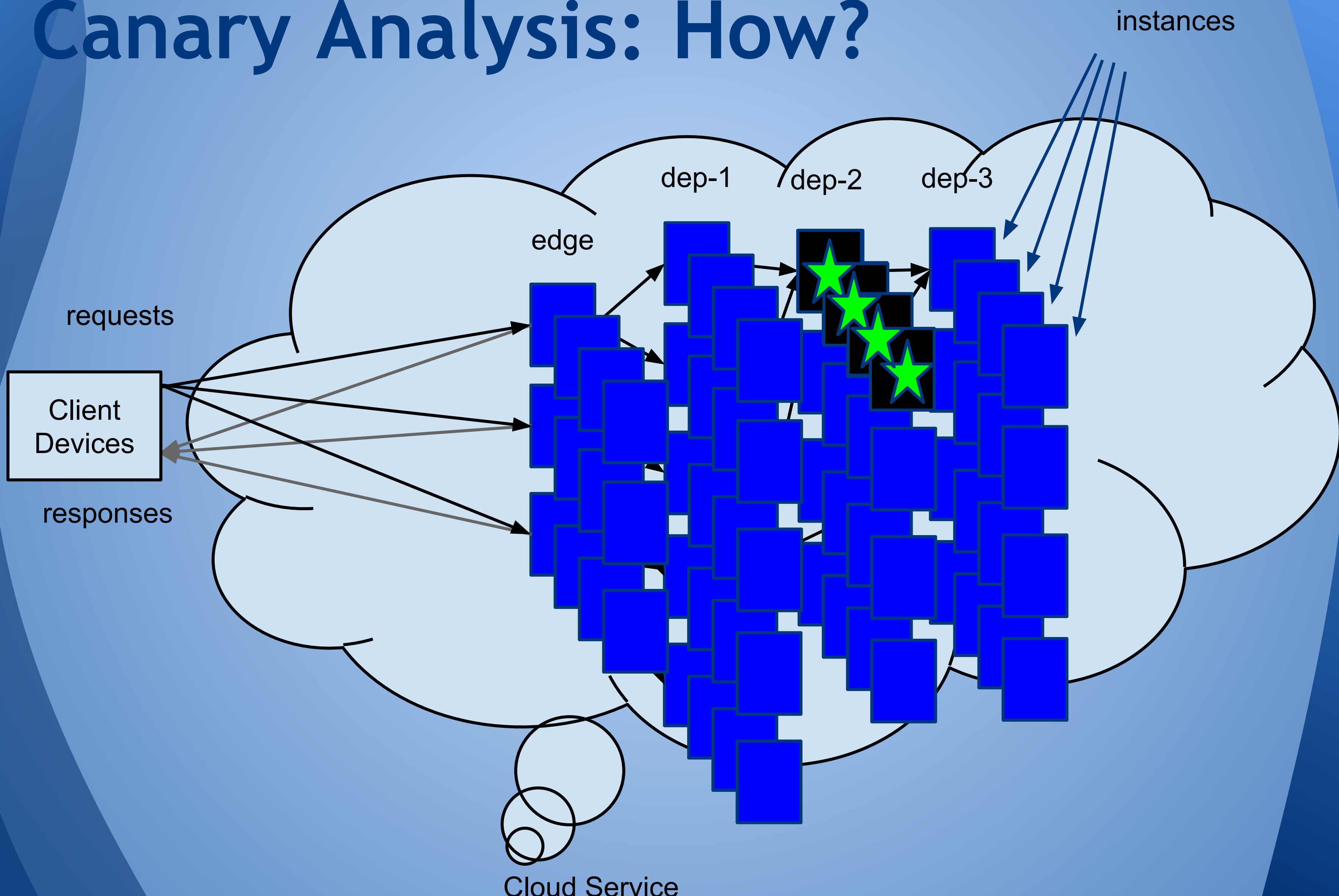
Canary Analysis Defined

- For a given service:
  - Deploy new code to limited #instances
  - Analyze against existing production code
  - Decide whether good or bad
  - Push forward (upgrade all service instances)
    - or roll back.

To Avoid This...



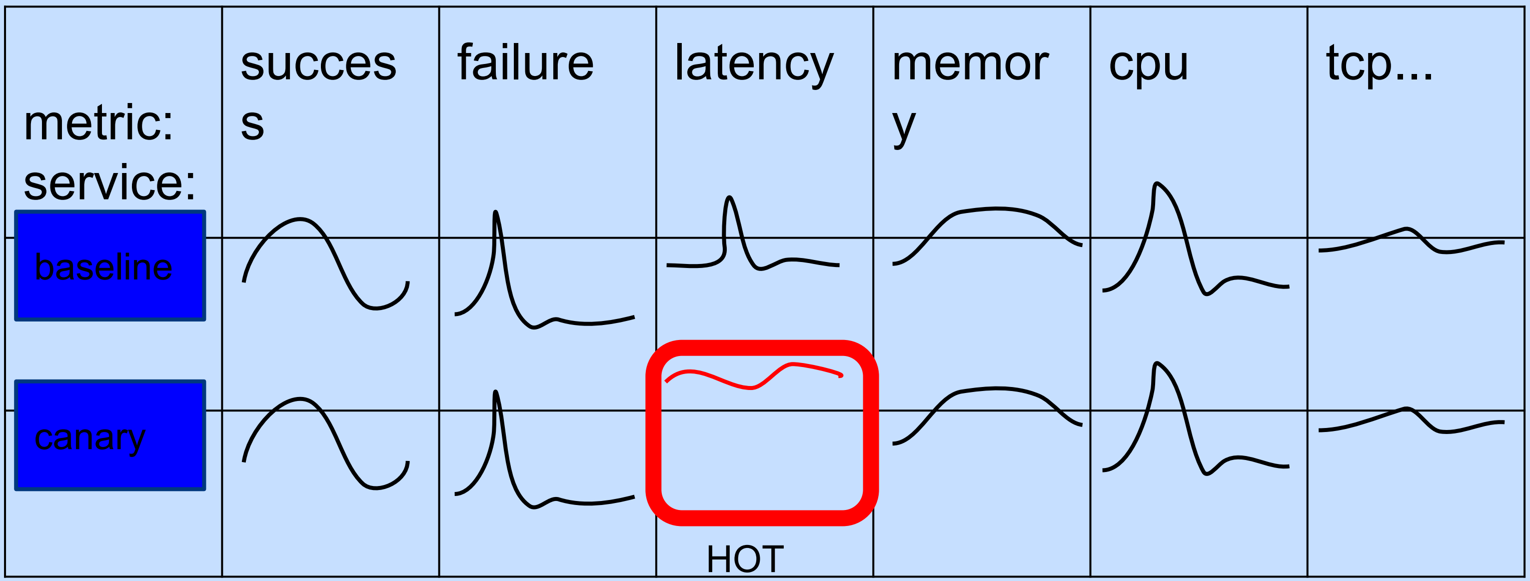
Canary Analysis: How?



Canary Analysis

- How does this work?
  - Service metric grid (again), 2 rows.
  - Compare canary to baseline, statistical tests.

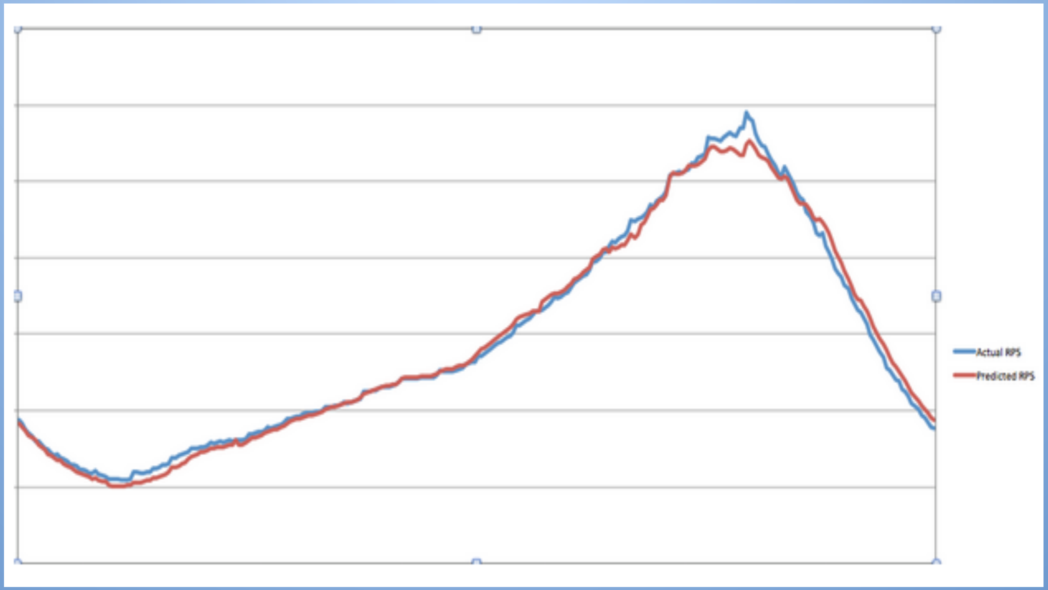
Automated Canary Analysis



Autoscaling

Load Based Autoscaling

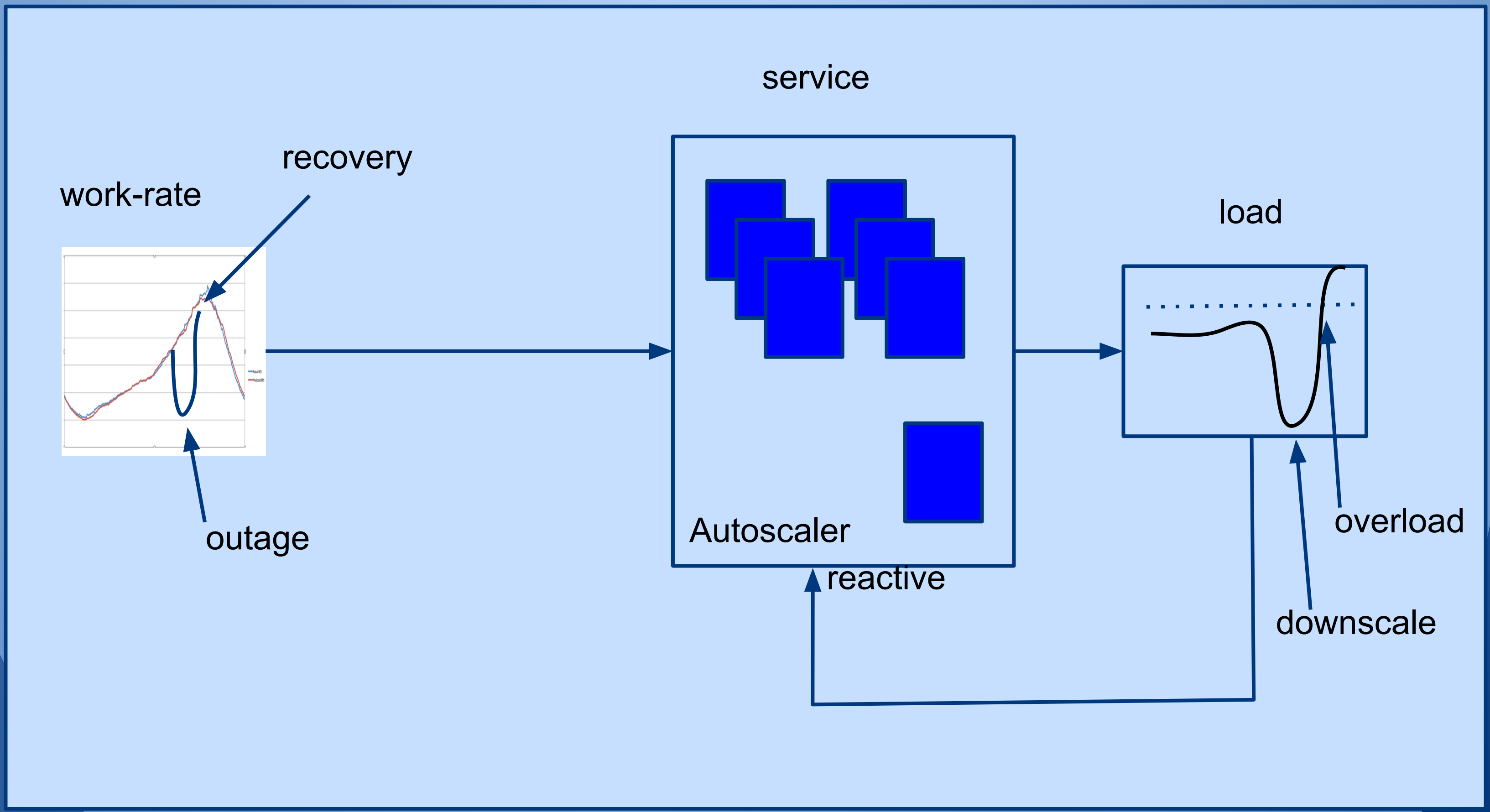
- increase #instances when load increases
- decrease #instances when load decreases
- works well...



Except when it doesn't..

- During an outage, load drops
- Instances are terminated
- Service becomes underprovisioned for return to normal request rate
- Overload occurs
- Other services suffer.
- Chaos.

Reactive Autoscaling

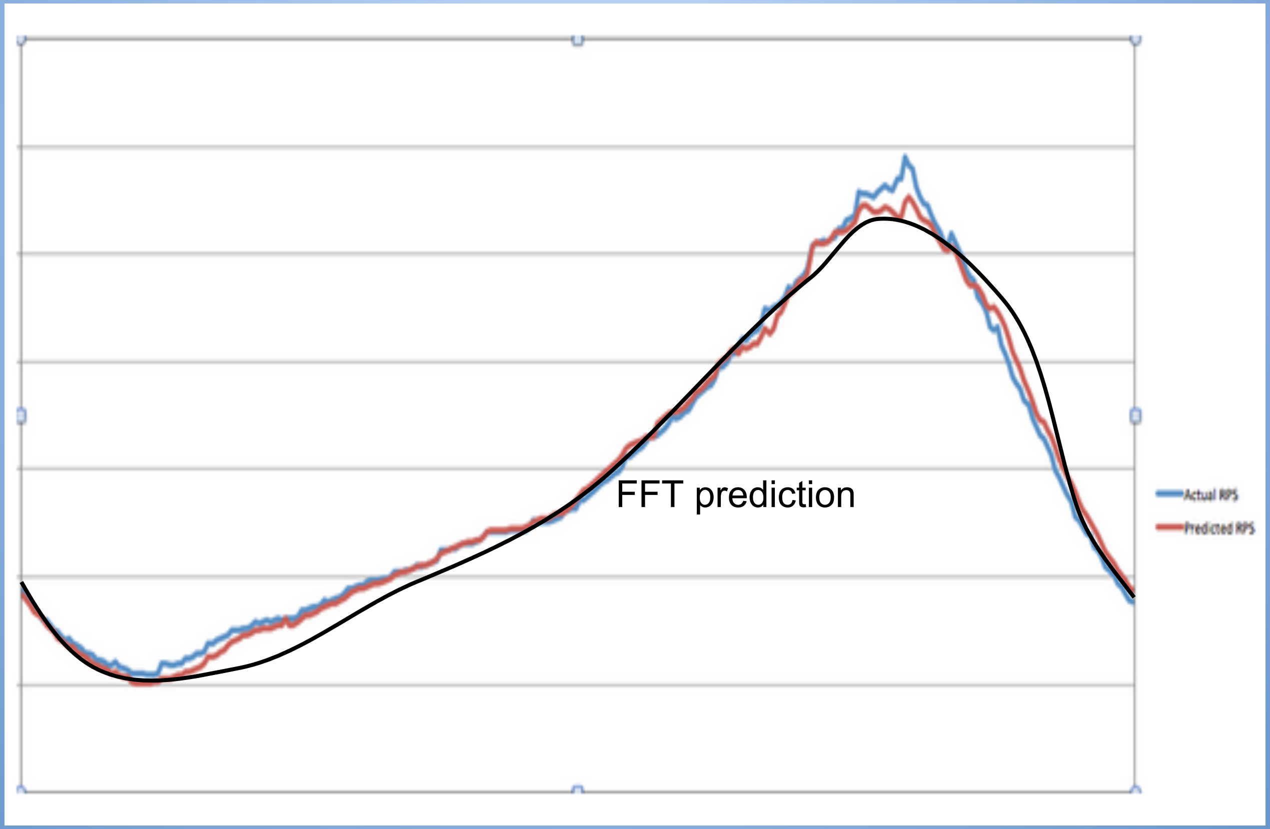


How do you avoid this?

- Use feedforward control
- Base on prediction of request rate
- Simple application of FFT low-pass filter.

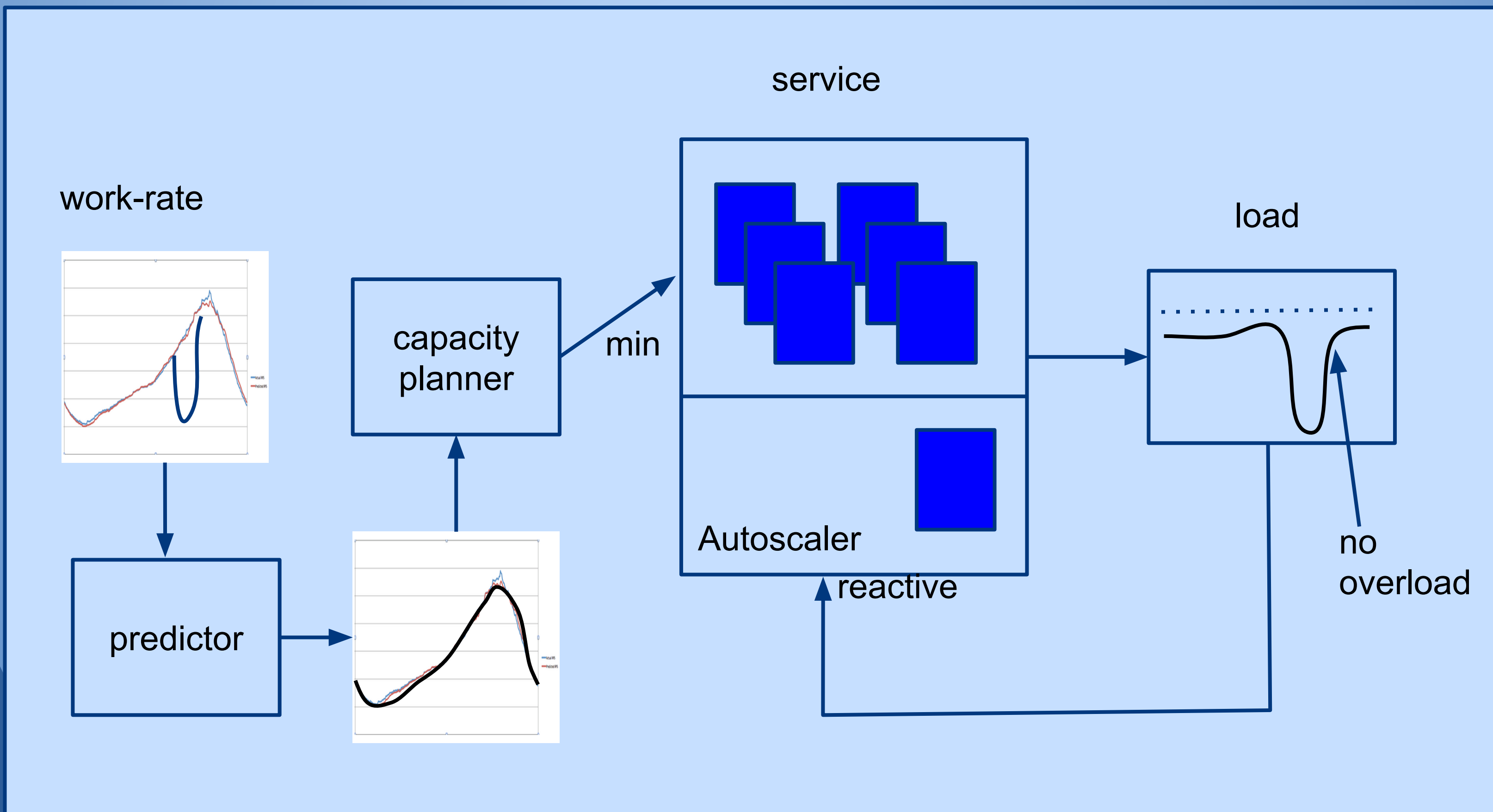
Scryer

- FFT based prediction



Netflix: Scryer

- Predictive+Reactive = Feedback Control



Real Time Analytics Engine

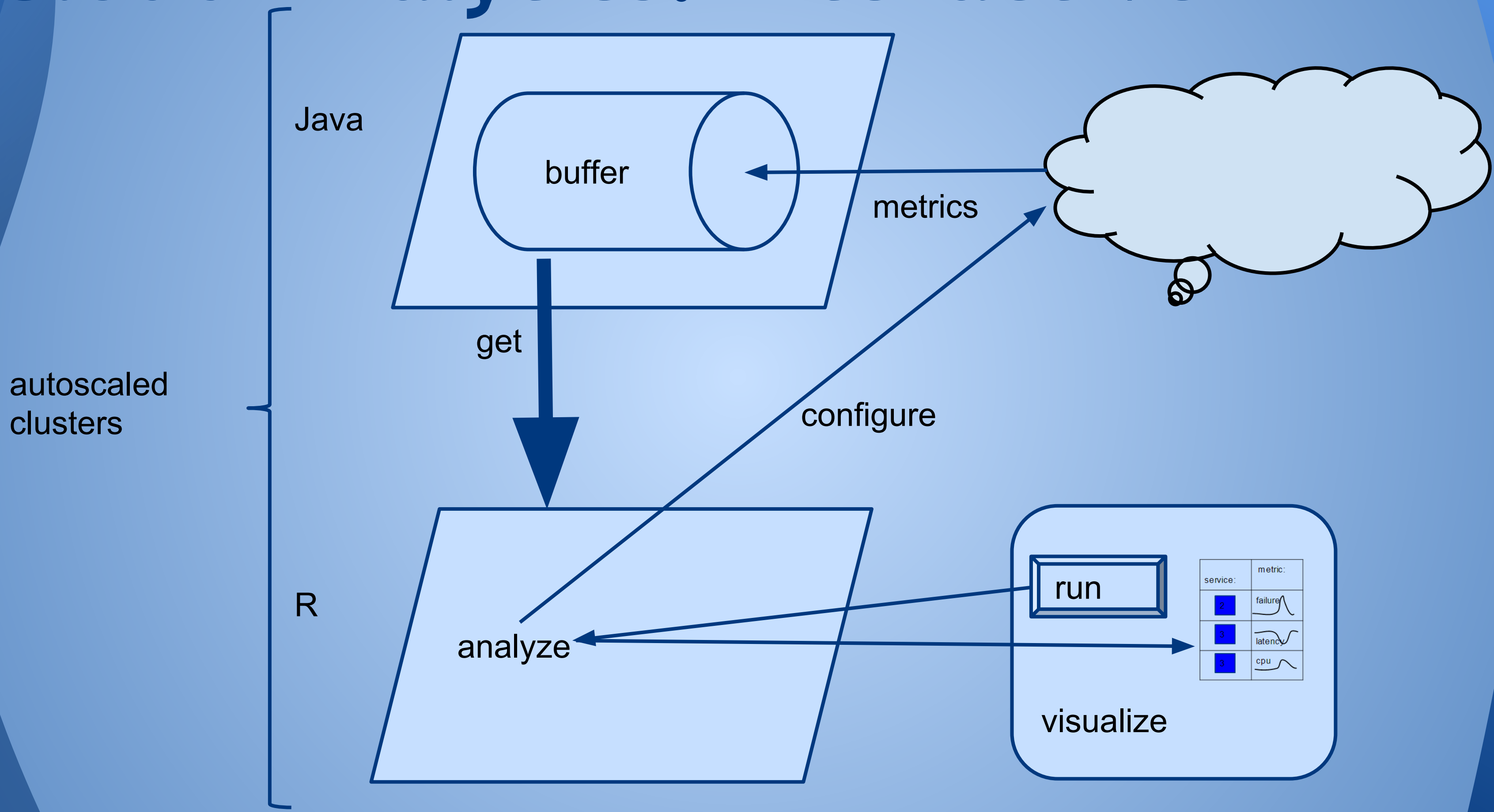
Analytics at Scale

- How do you do analytics at scale?
  - Do monitoring at scale
  - Do data-collection & buffering at scale
  - Run Analytics at scale
  - Use the Cloud to achieve scale.
- (But use a different Cloud).

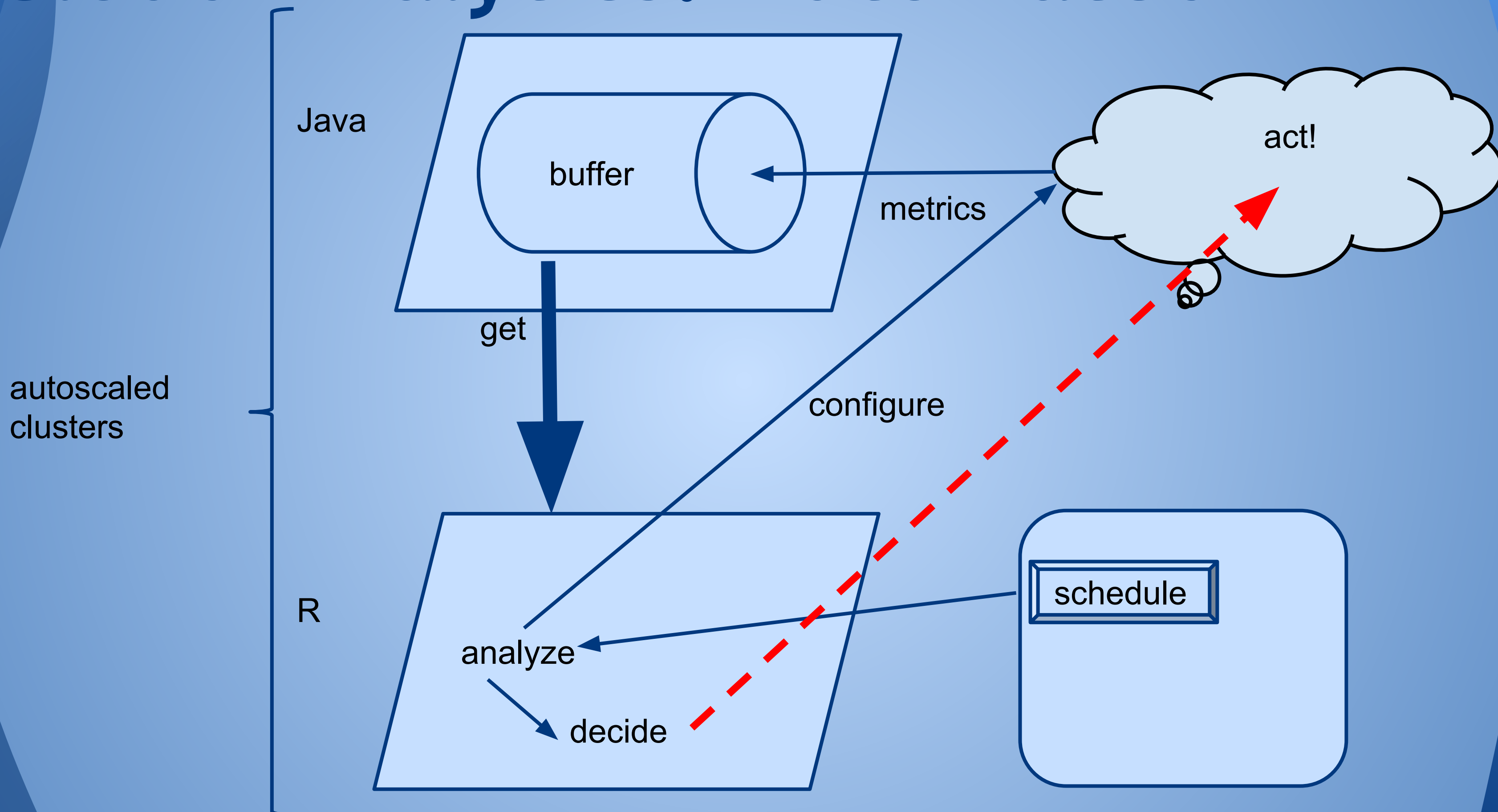
Analytics at Scale

- One possible architecture: Java and R engines in the Cloud
  - gathering data
  - running analytics
  - performing visualization
  - doing notification

Cloud Analytics: Interactive



Cloud Analytics: Automated



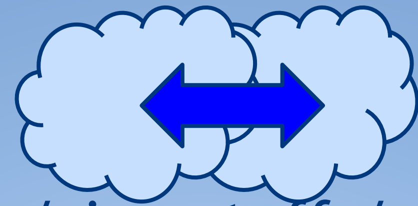
Analytics Challenges

Cloud Analytics: Big Challenges

- instance outlier detection at scale
- tuning queues & timeouts for services
- detection of overload/underprovision
- anomaly detection (prediction)
- behavior pattern classification
- automatic alert tuning
- “closing the loop”

“Cloudstream”

Cloudstream



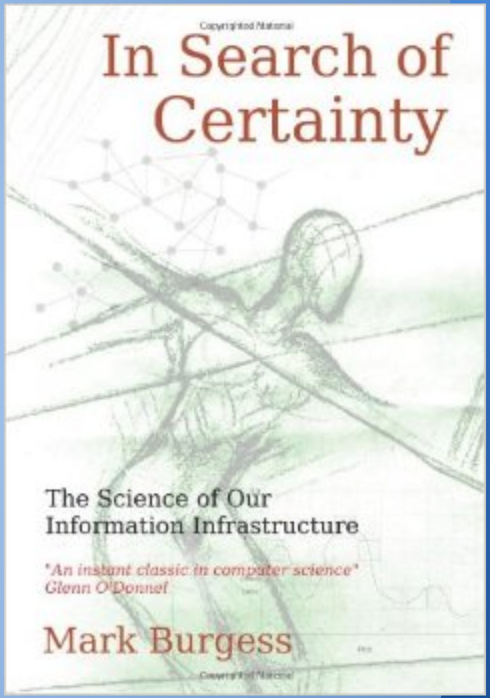
https://github.com/simontuffs/cloudstream/wiki

- Cloudstream Stack:
  - Netflix OSS, Open/CPU, iPython, Cloudsim, Amazon/Kinetics Netflix/Suro Storm/Spark
- Real-Time Analytics, in the Cloud, for the Cloud.
  - Currently building an application simulator
  - Design & train analytics

Questions?

Recommendation:

- Mark Burgess
  - In Search Of Certainty, 2013
  - Views information systems from a physics perspective, showing the non-deterministic complexity we are creating, and how hard it is to manage



Caution!

- Please seek a second opinion before spending years building a Ph.D. out of the following speculations & observations….

A Posteriori Observations

- Focus on $ not KWh for allocation
  - (they are isomporphic)
  - $ drive customer behavior the right direction
- Consider standardizing on “Model Predictive Controls” (e.g. GPC)
  - Superset all other linear methods, save time :)
- Most of my challenges do not close any control loops
  - other than estimation/modeling loops

A Posteriori Challenges

- Monitoring Validation
  - Our Cloud is down! Our Monitoring is down!
  - How can you tell?
- Avoid WOM (write-only monitoring)
  - how to aggregate useful data without losing information but still do analytics
- Causality
  - Infer dependency graph from data?
  - Cross-covariance for causation.

A Posteriori Challenges

- Develop Cloud invariants/assertions as "models of behavior"
  - increased latency => upstream errors
  - upstream errors => downstream request drop
  - increased cpu => increased latency
  - increased requests => increased (cpu, load)
  - parameterize & tune a behavioral model base on these invariants.

A Posteriori Challenges

- Machine learning (SVM, markov models)
  - Behavioral classification
  - Failure identification
- Evidence based learning
  - Bayesian networks for fault detection.
- Better predictors
  - Wavelets, basis functions.
- Modeling the Cloud
  - Dynamic Equilibrium
  - Transient Dynamics
  - “Kalman” Filtering

A Posteriori Challenges

- Auto-tune configuration parameters *(close the loop)*
  - 99.5% latency ⇔ errors => need to increase caller timeouts.
  - 99.5% latency ⇔ load => need to scale up if at the “knee”.
  - 99.5% queue size ~ max-size => need to add worker threads
  - do this in production, across operating ranges

Thankyou!