Anomaly Detection
Fault Tolerance
Anticipation

Patterns

John Allspaw
SVP, Tech Ops
Qcon London 2012

Etsy

Four Cornerstones

Erik Hollnagel

(Anticipation)

Knowing What To Expect

Knowing What To Look For

(Monitoring)

(Response)

Knowing What To Do

Knowing What Has Happened

(Learning)

Four Cornerstones

Erik Hollnagel

(Anticipation)

(Response)

Knowing What To Expect

Knowing What To Look For

Knowing What To Do

Knowing What Has Happened

(Monitoring)

(Learning)

Four Cornerstones

Erik Hollnagel

(Anticipation)

Knowing What To Expect

Knowing What To Look For

(Monitoring)

(Response)

Knowing What To Do

Knowing What Has Happened

(Learning)

Anomaly Detection

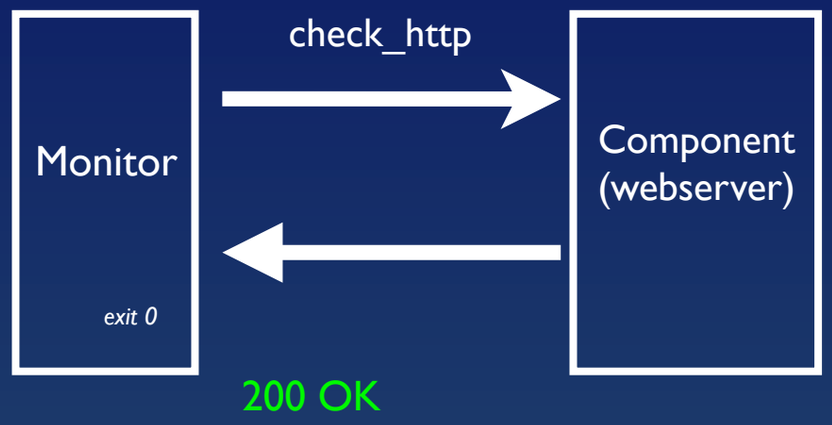
Anomaly Detection

- Getting at the state of health
- Evaluating the state of health
- Components AND systems

Supervisory

Example: Active health check

Supervisory



Pros:

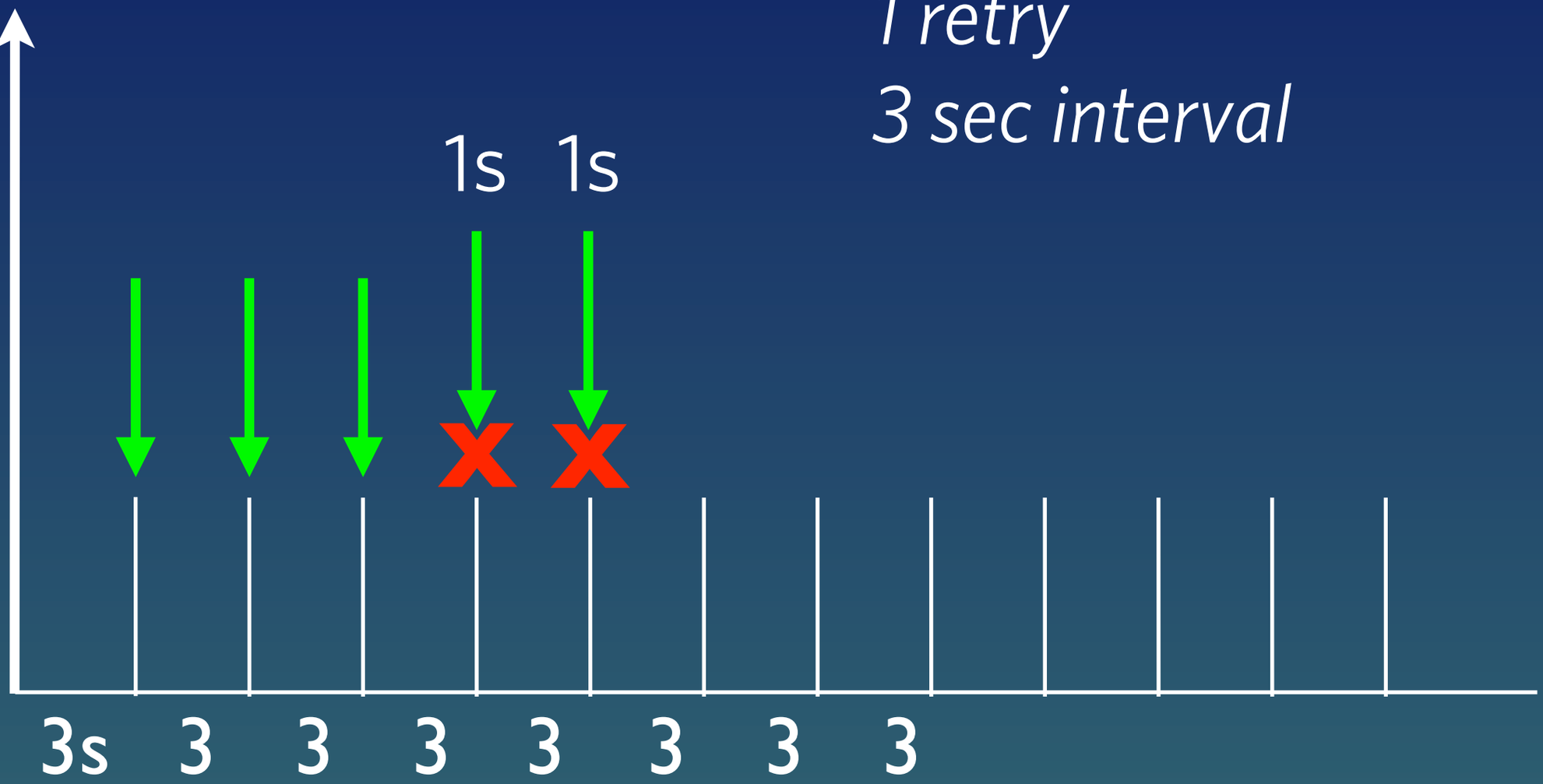
- Easy to implement
- Easy to understand
- Well-known pattern

Cons:

- Messaging can fail
- Scalability is limited

Supervisor Sensitivity

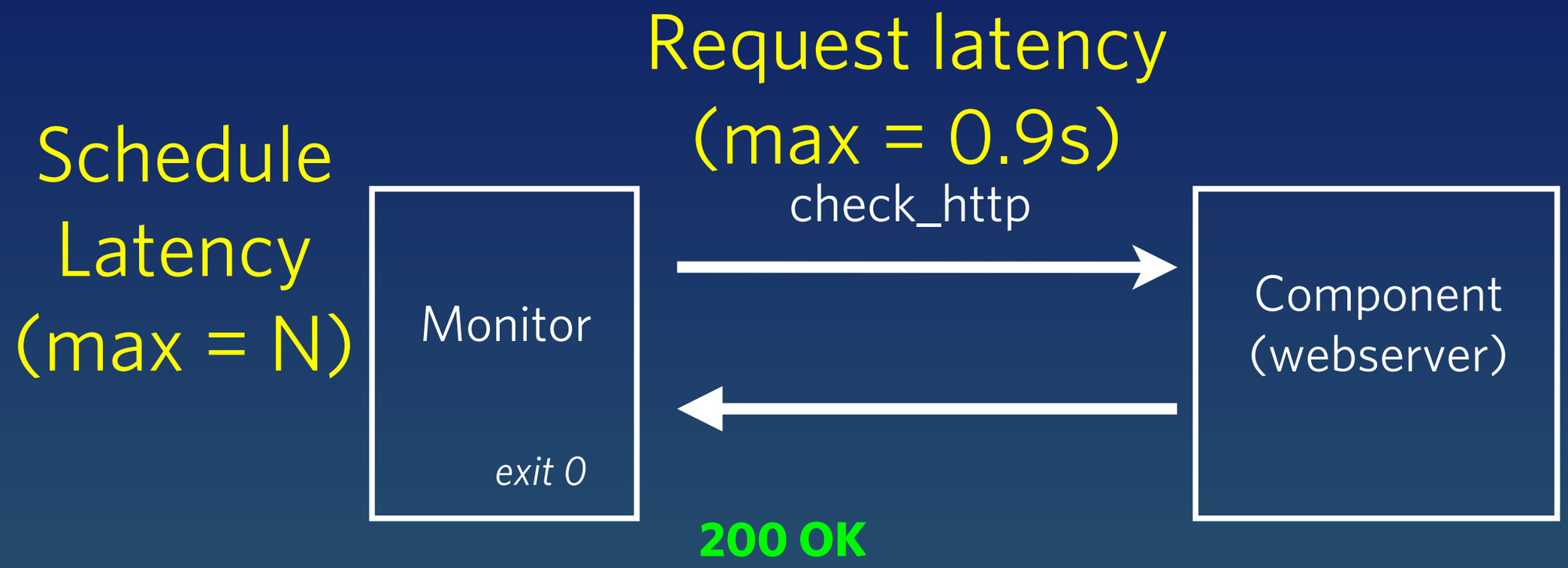
1 sec timeout
1 retry
3 sec interval



(7.9 sec exposure)

Up to ~2.9s for the previous interval

Supervisor Sensitivity



Supervisor Sensitivity

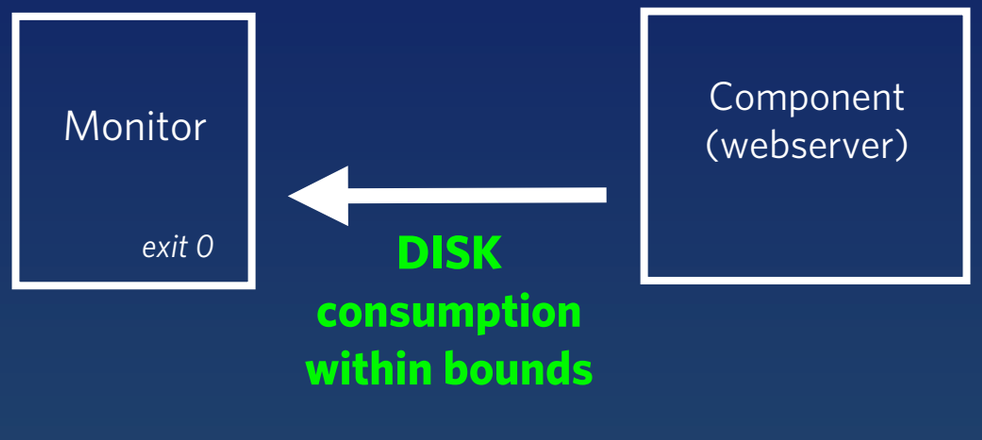
How many seconds of errors can you tolerate serving?

Supervisory

Example: Interval Passive health check

Supervisory

Example: Interval Passive health check



Pros:

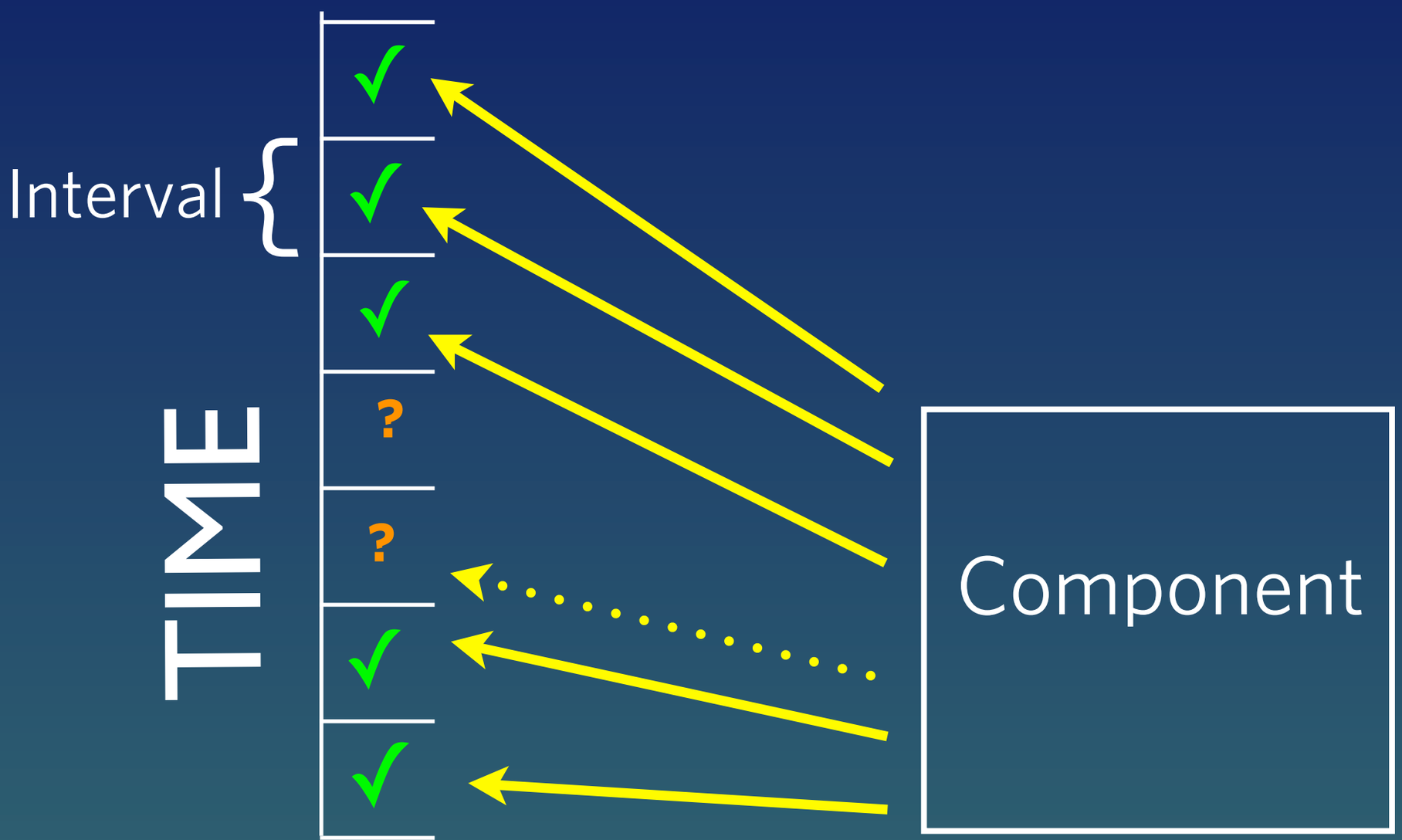
- Efficient
- Scalability is different
- Fewer moving parts
- Less exposure
- Can submit to multiple places

Cons:

- Nonideal for network-based services
- Different tuning (windowed expectation)

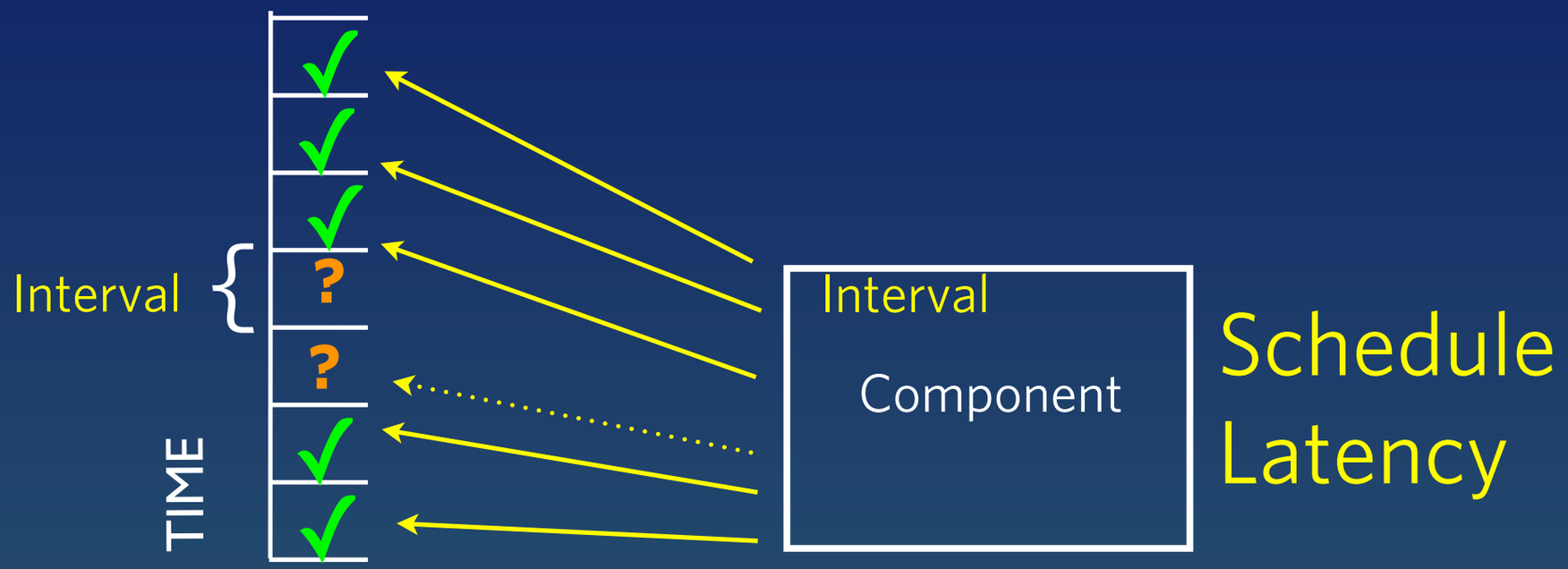
Supervisory

Example: Passive health check



Supervisory

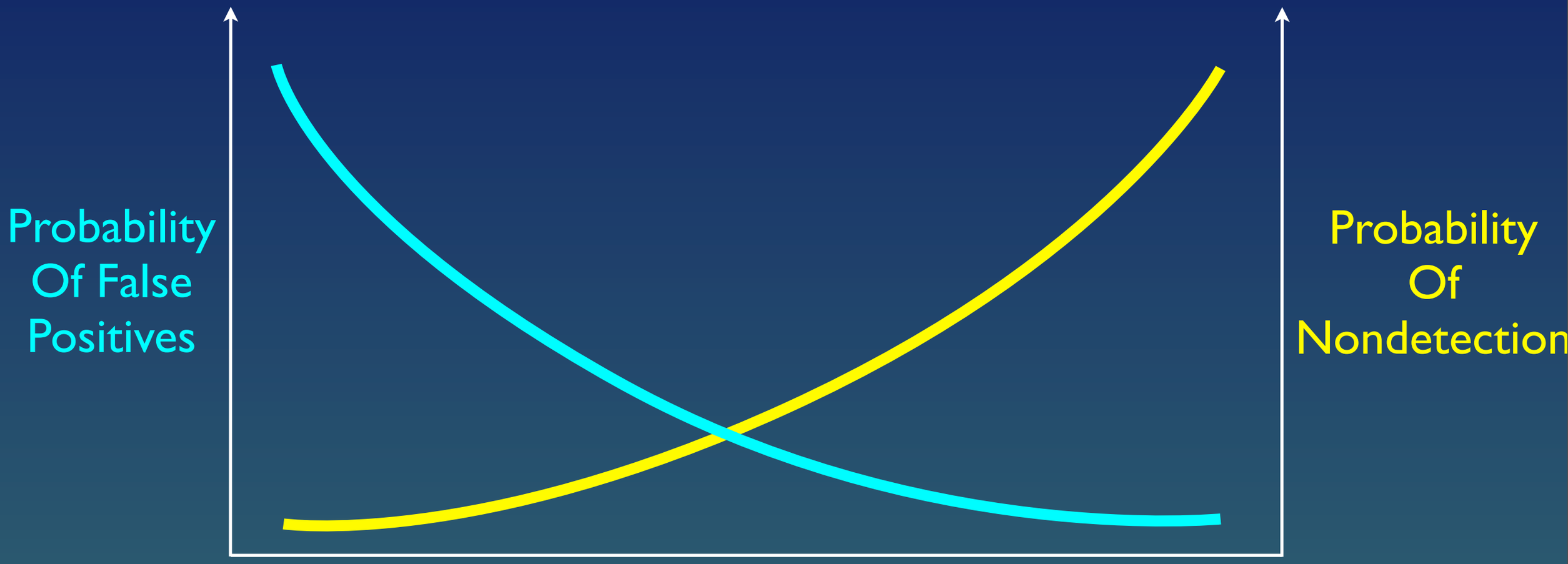
Example: Passive health check



Exposure =

(Schedule + Interval)*UnknownConsecutiveIntervals+1

Frequency and Transience

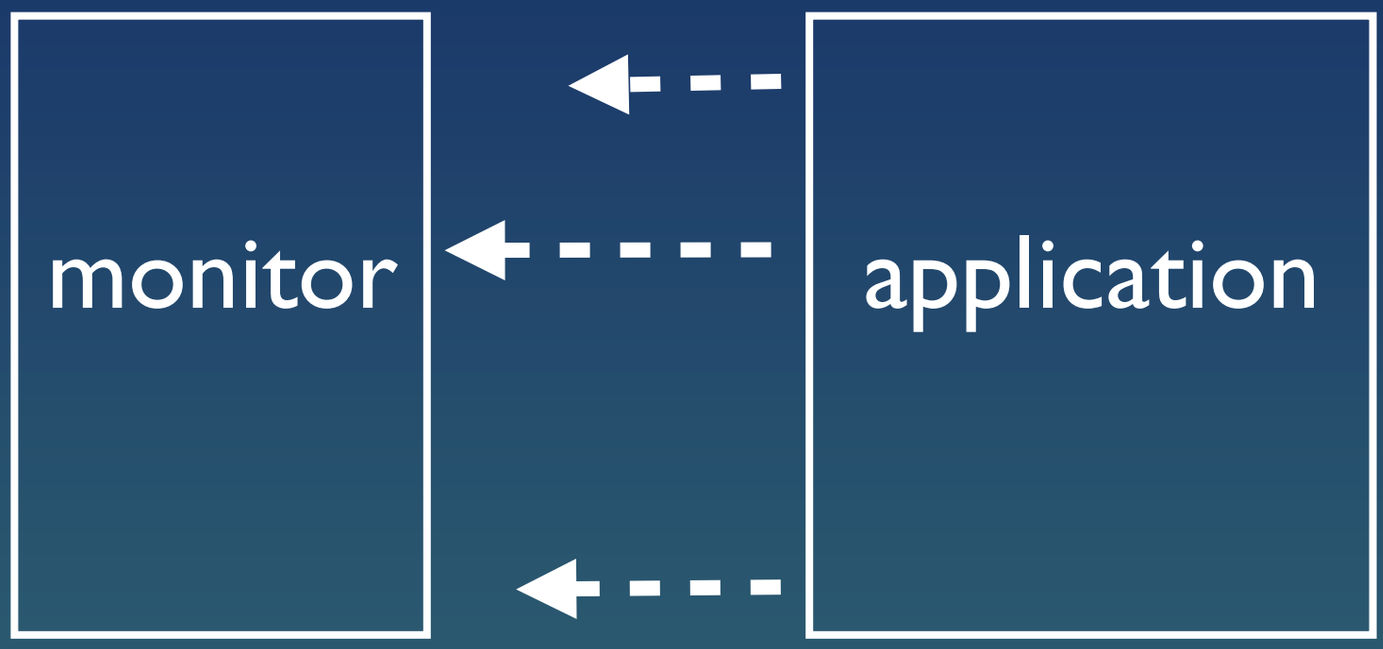


Short intervals
Low # of retries
Short timeouts

Long intervals
High # of retries
Long timeouts

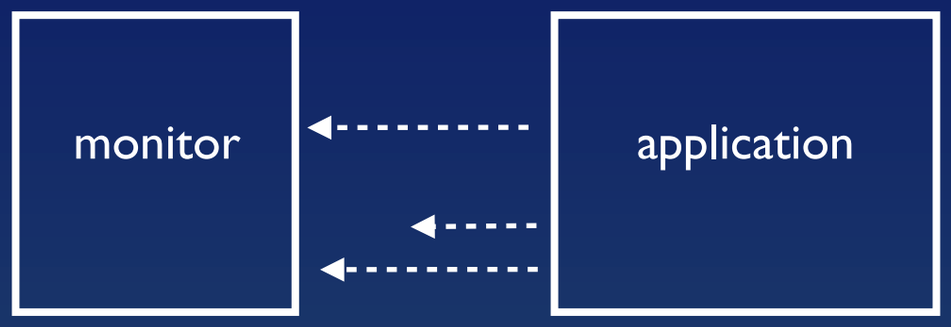
In-Line

Example: Passive application event logging



Supervisory

Example: Passive application event logging



Pros:

On-demand publish

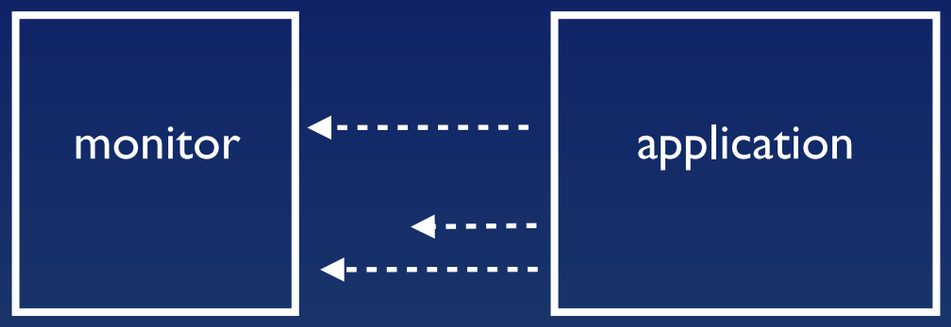
Cons:

Onus is on the app

Can't be 100% sure it's working

Supervisory

Example: Passive application event logging

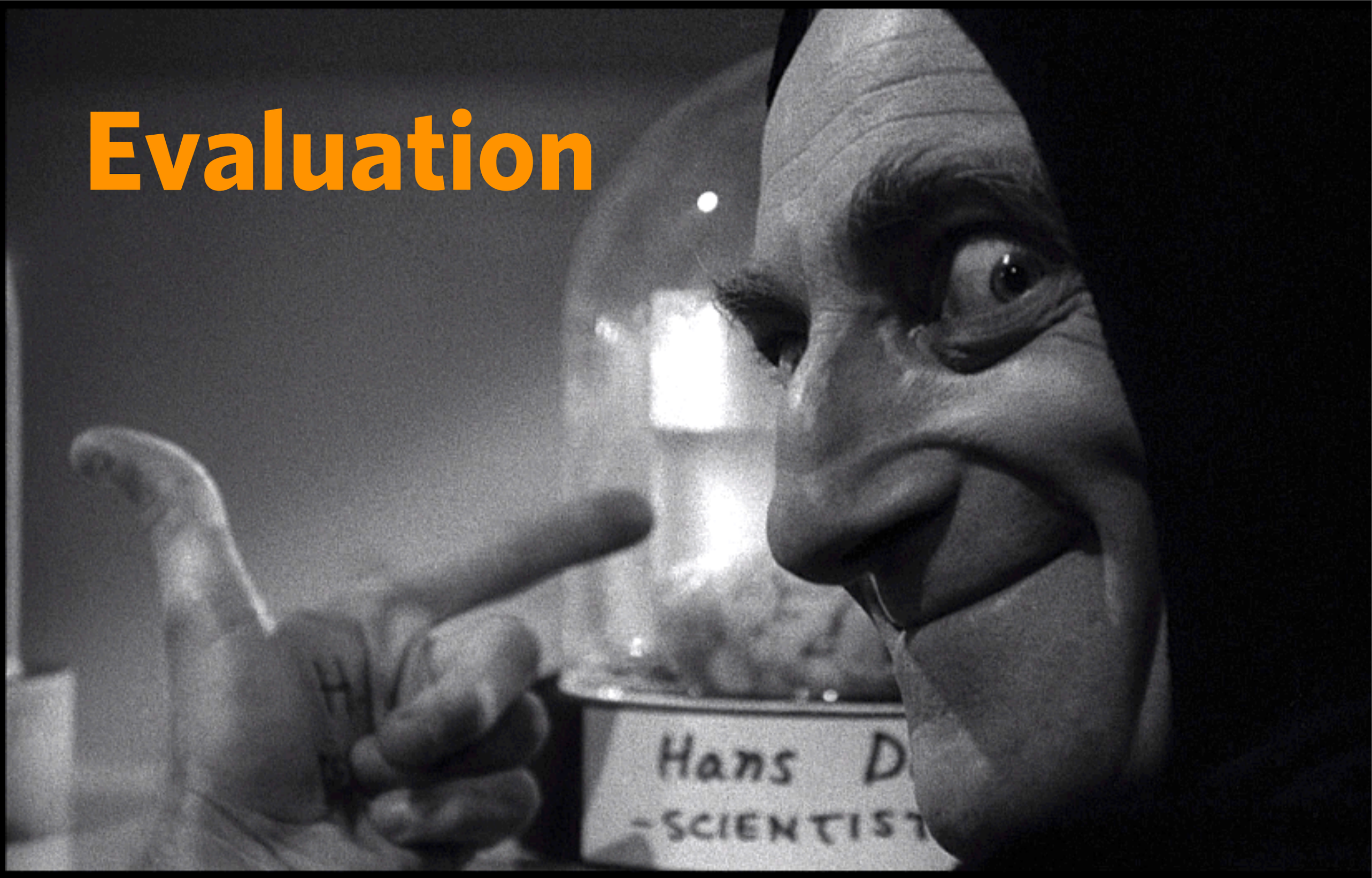


Positive events (sales, registrations, etc.)
Negative events (errors, exceptions, etc.)

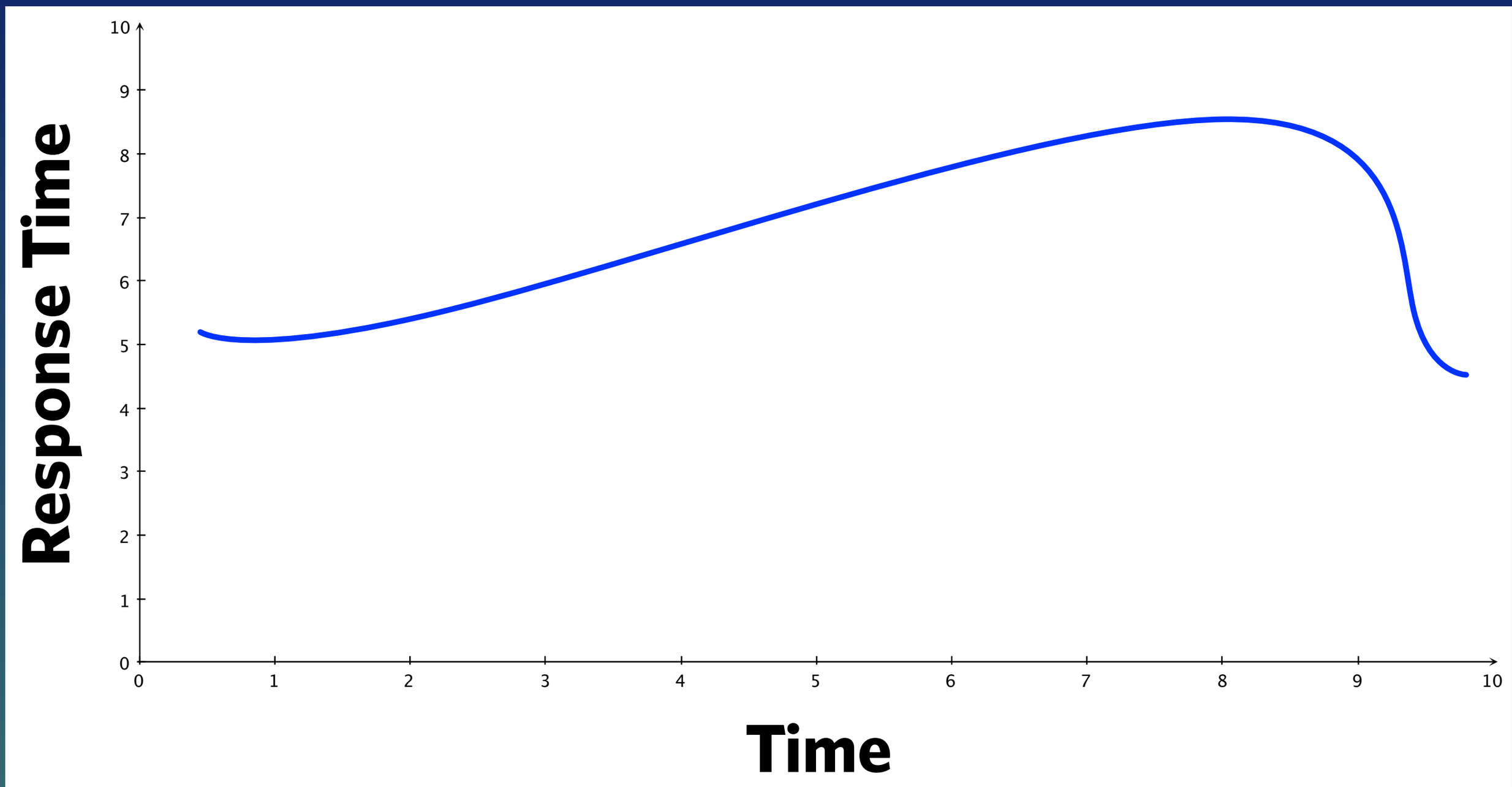
Lack or presence of data mean different things, so history is paramount.


Context

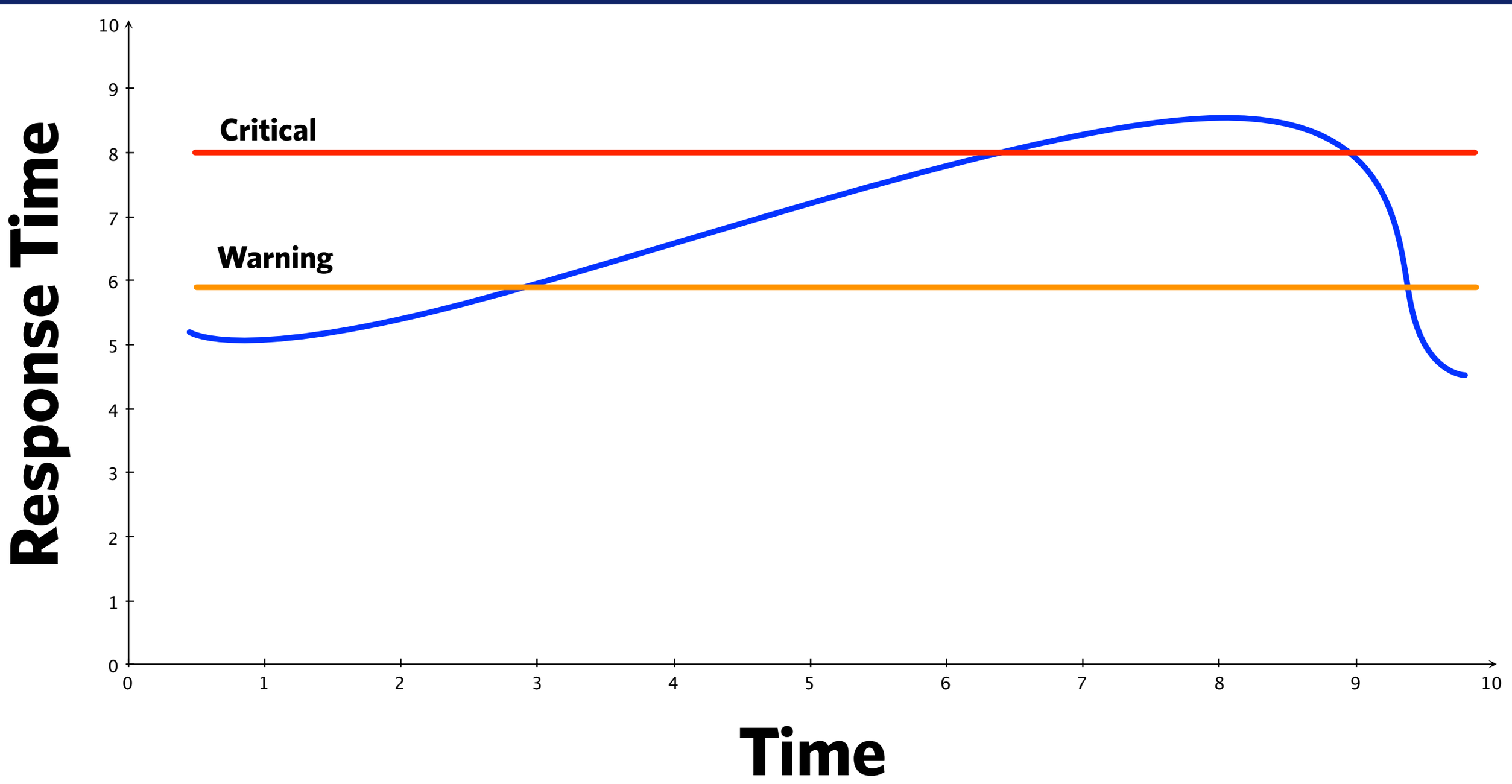
Evaluation



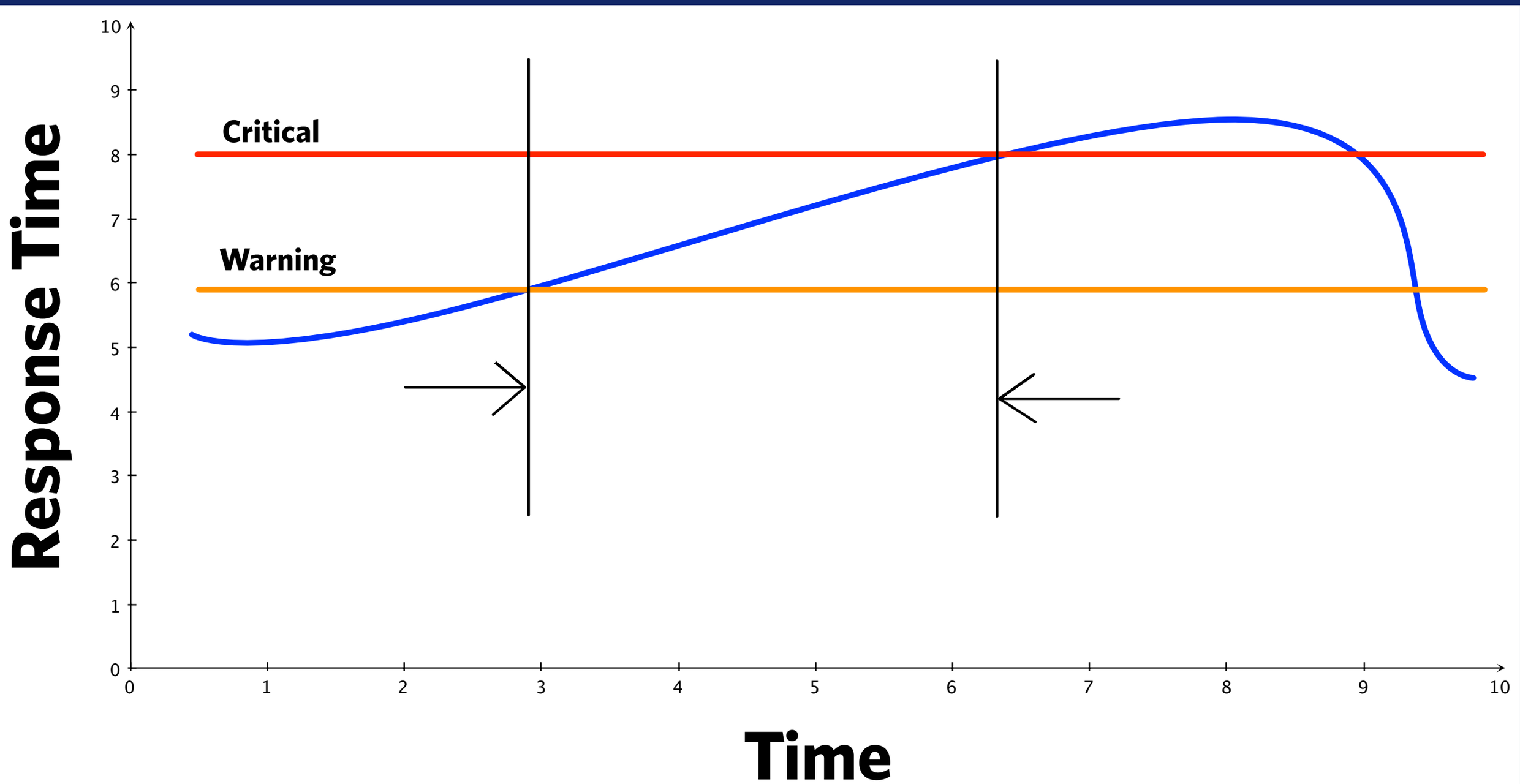
what is ‘abnormal’ ?


Response Time
Time
0
1
2
3
4
5
6
7
8
9
10

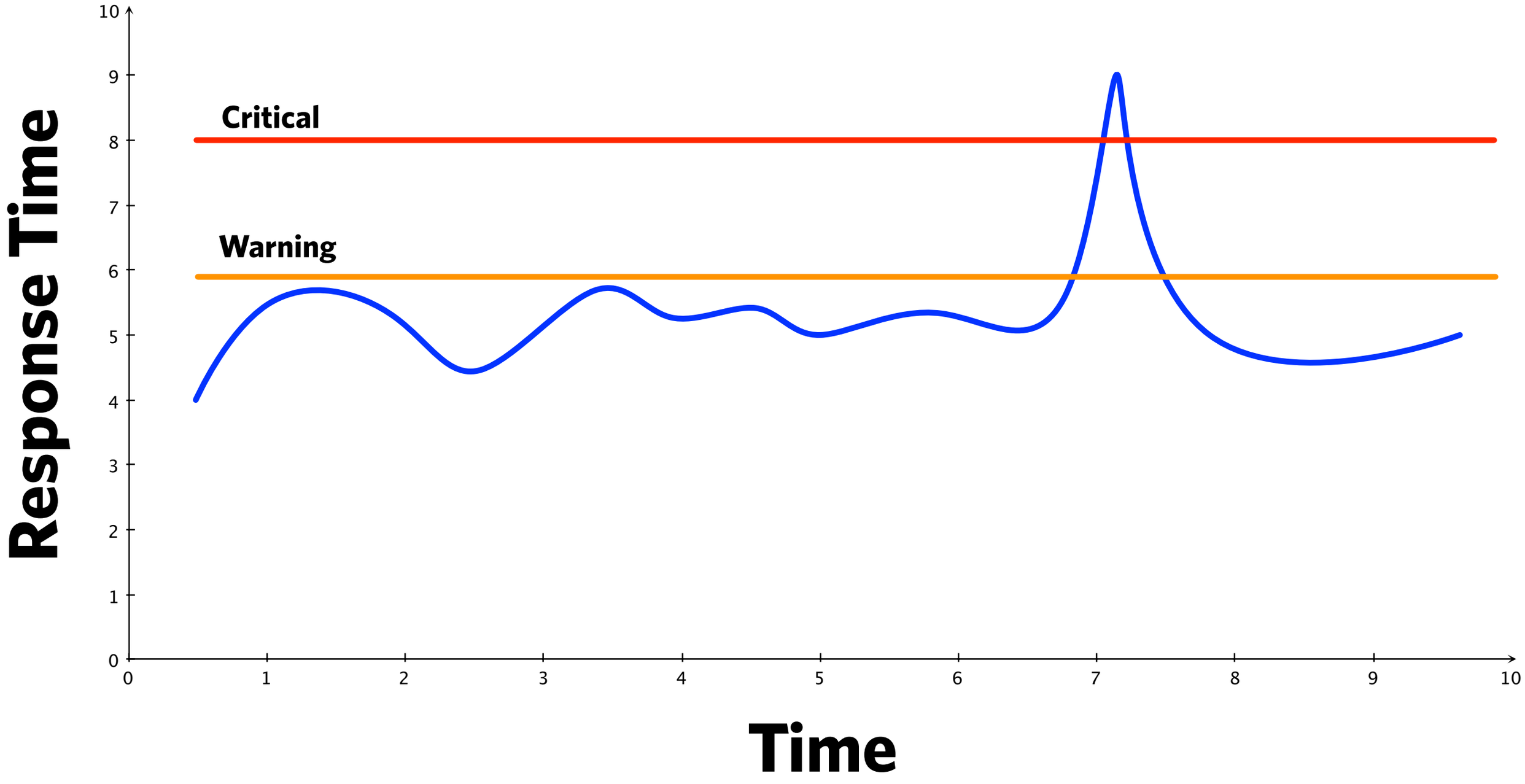
Static Thresholds



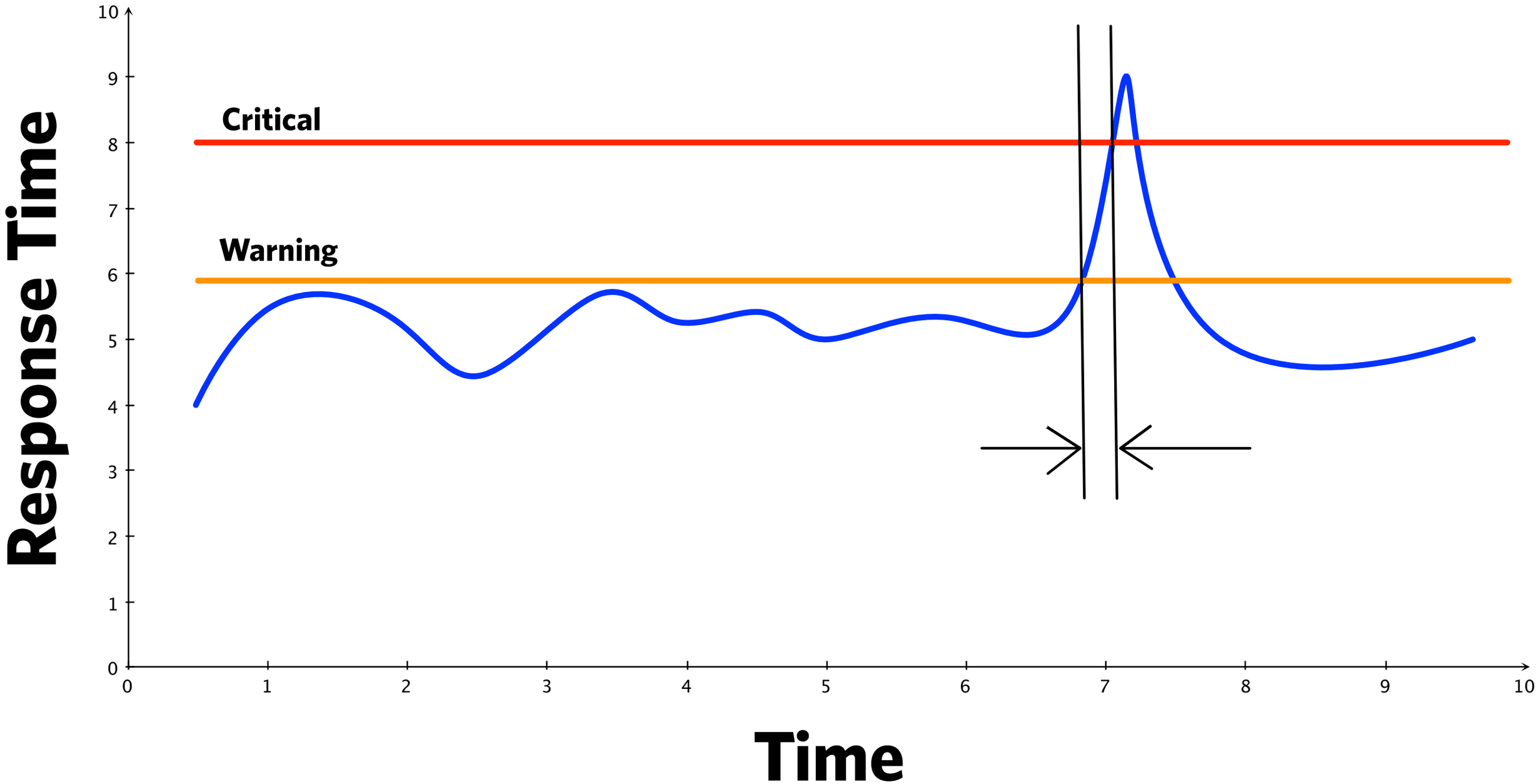
Static Thresholds



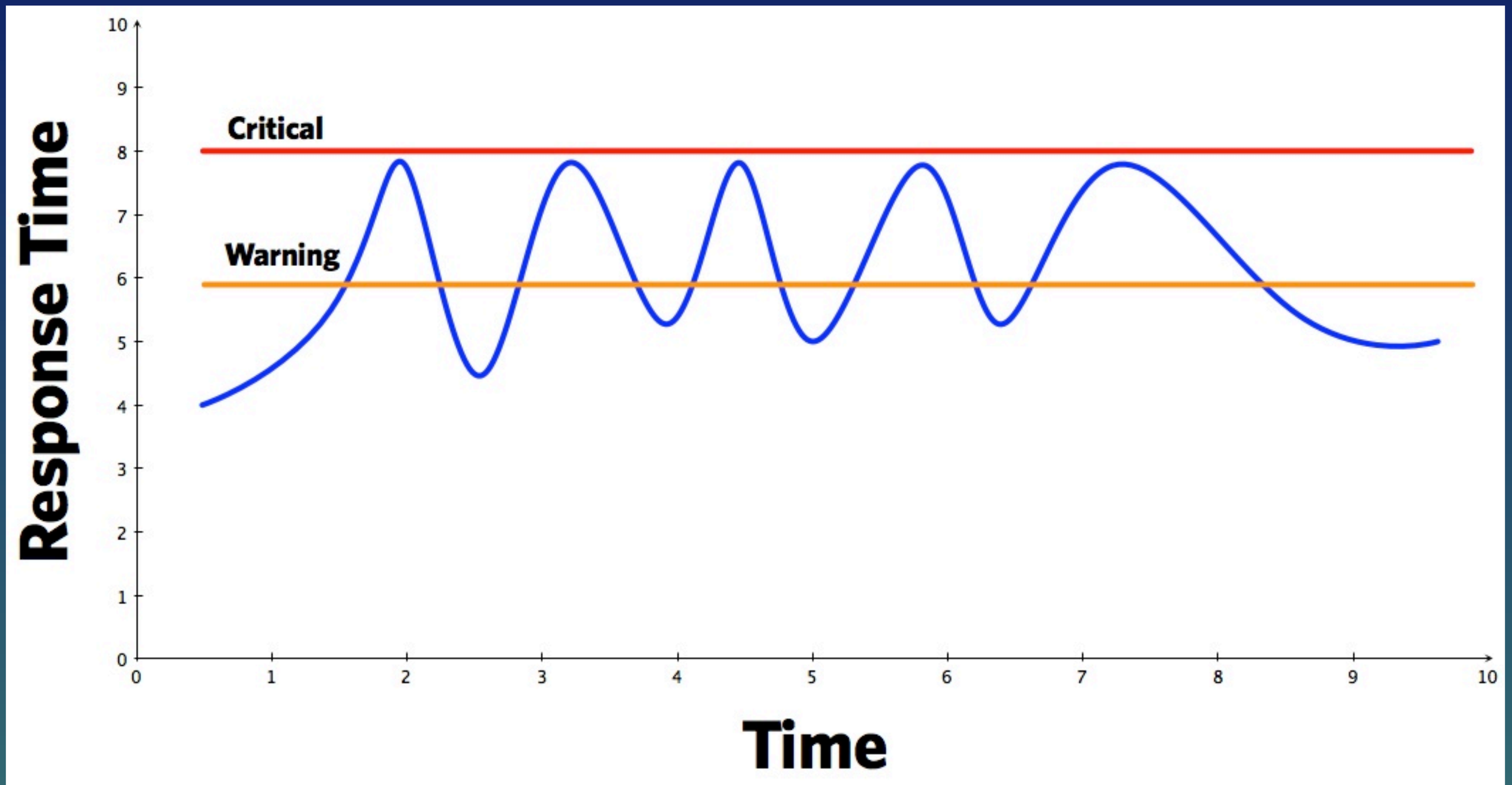
Static Thresholds



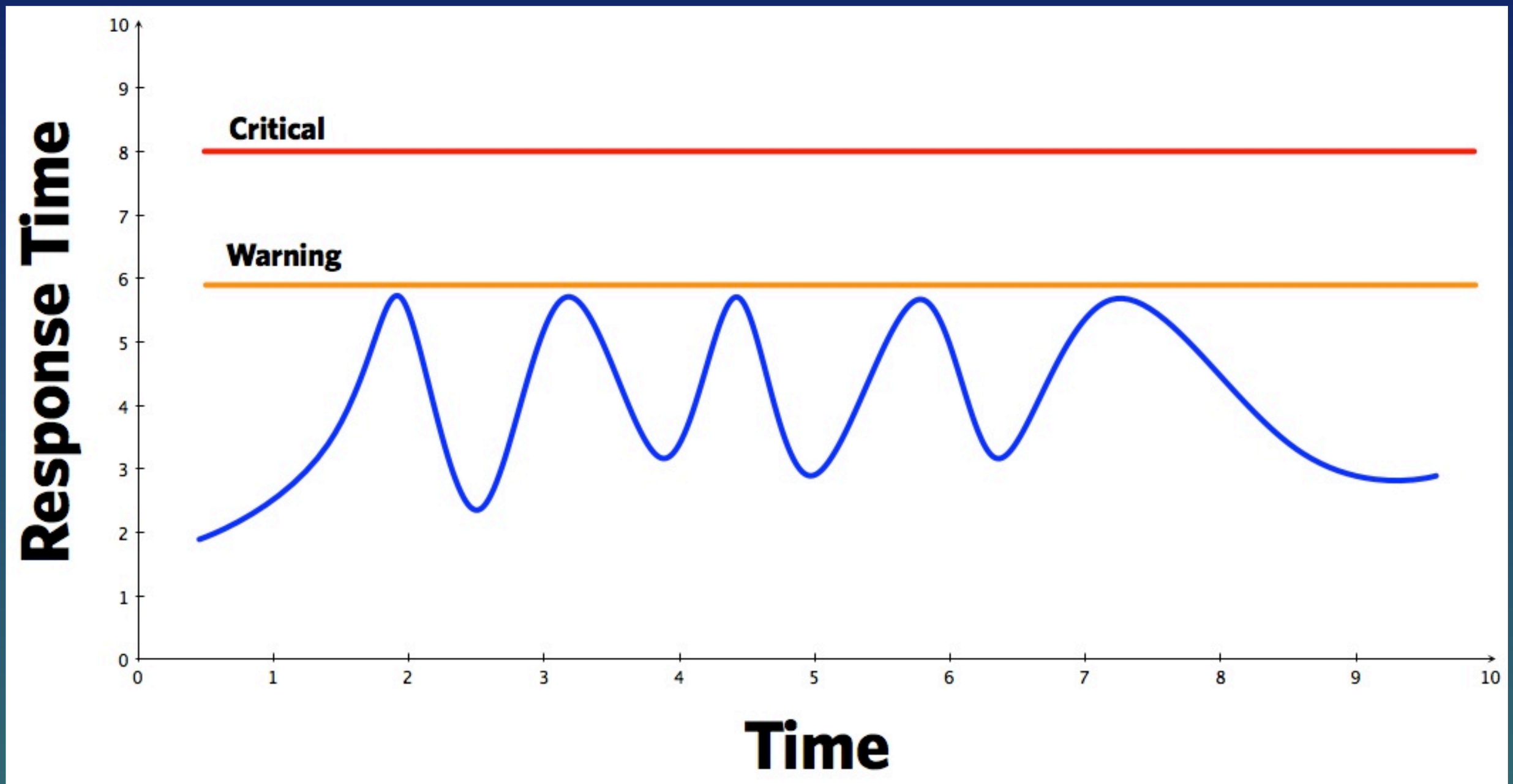
Static Thresholds



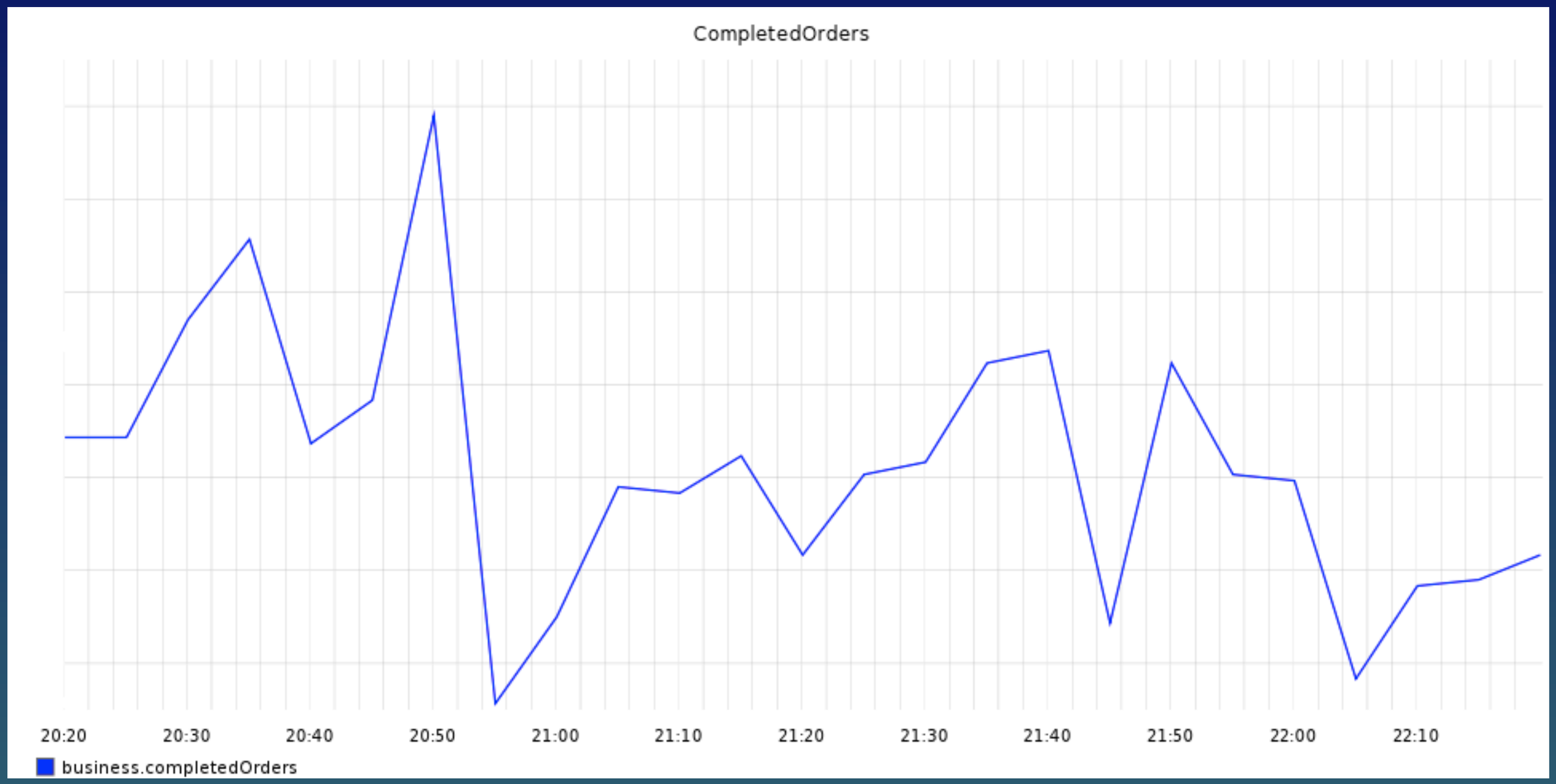
Static Thresholds



Static Thresholds

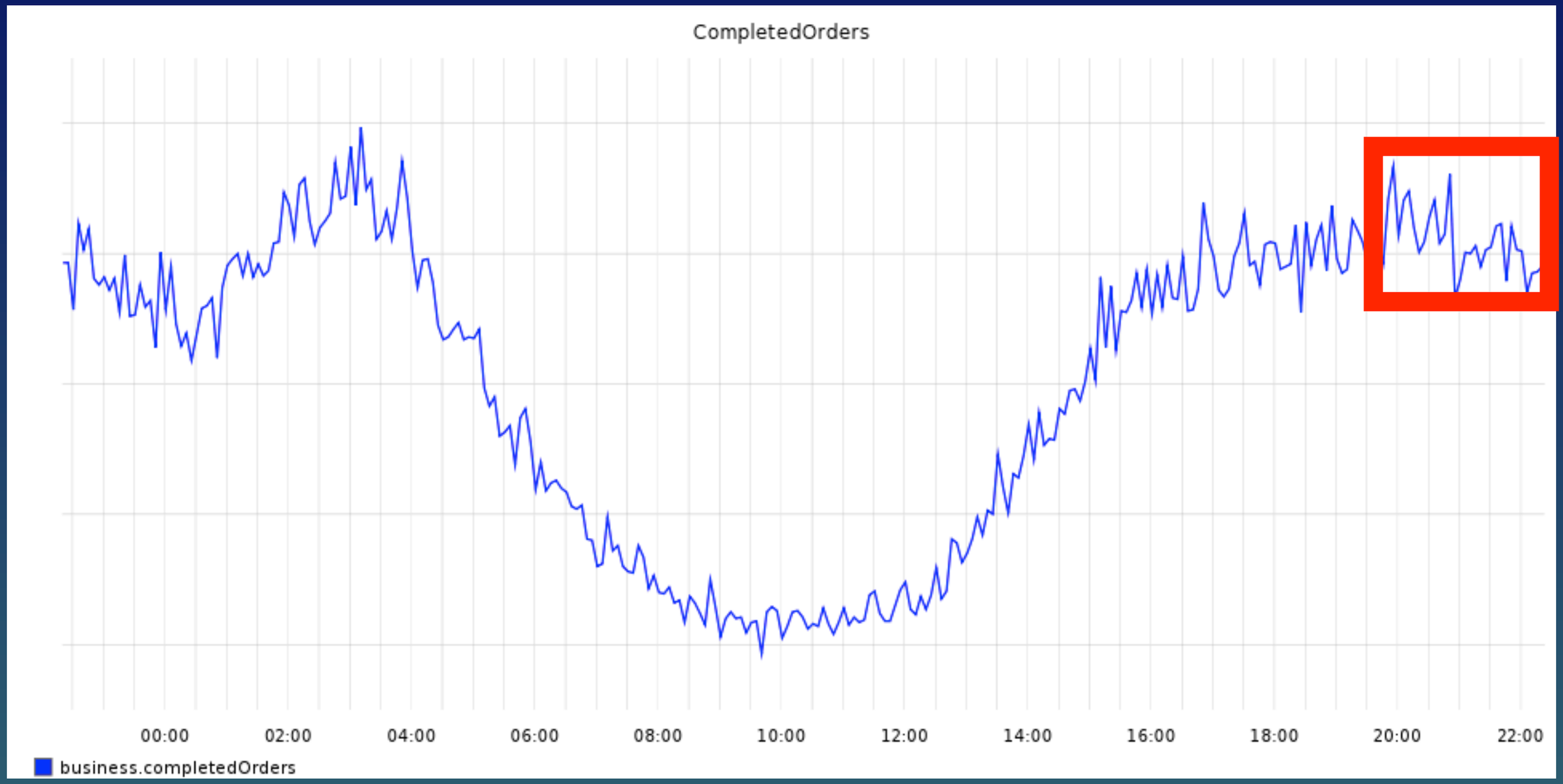


Context



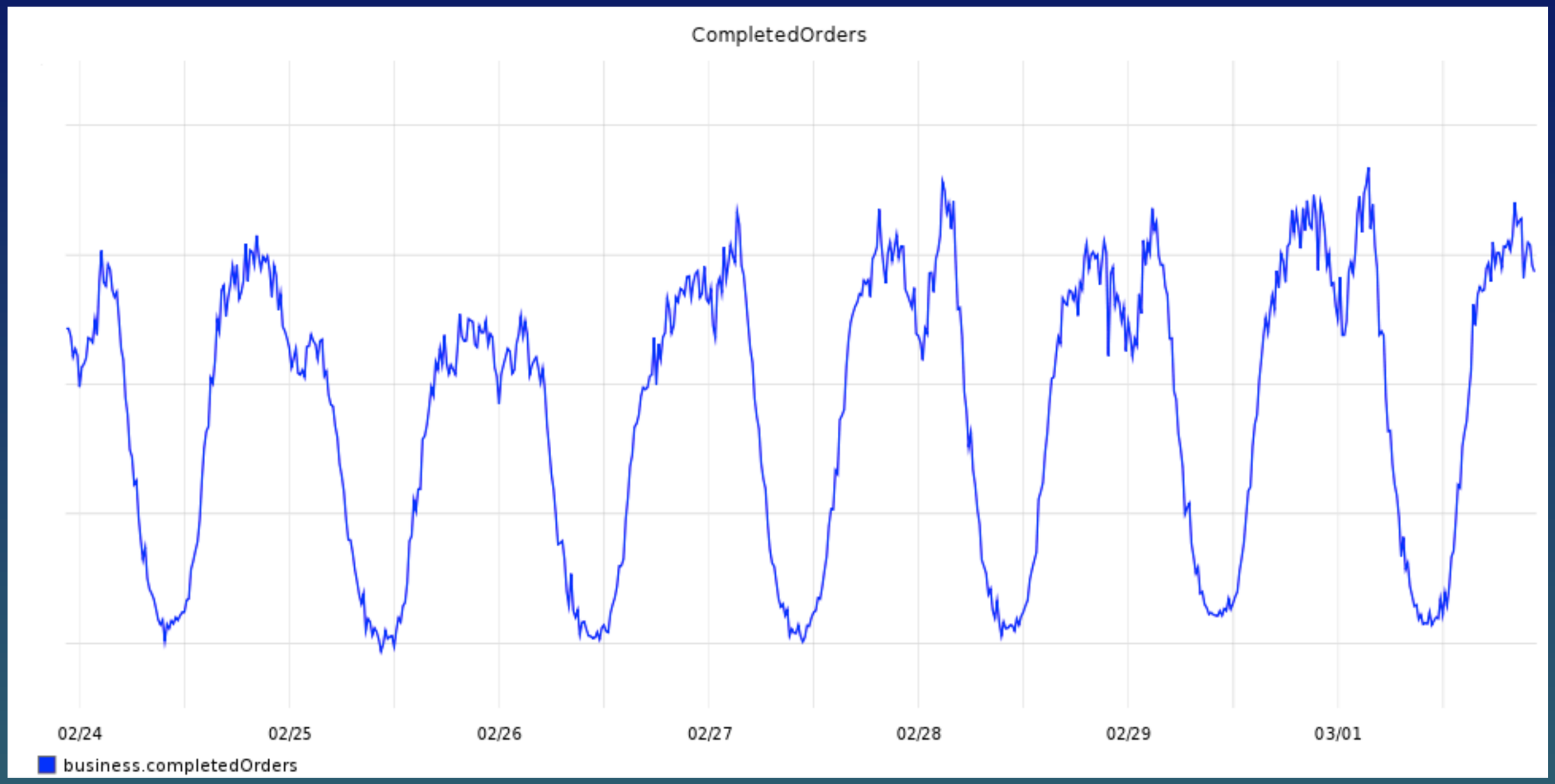
Normal?

Context



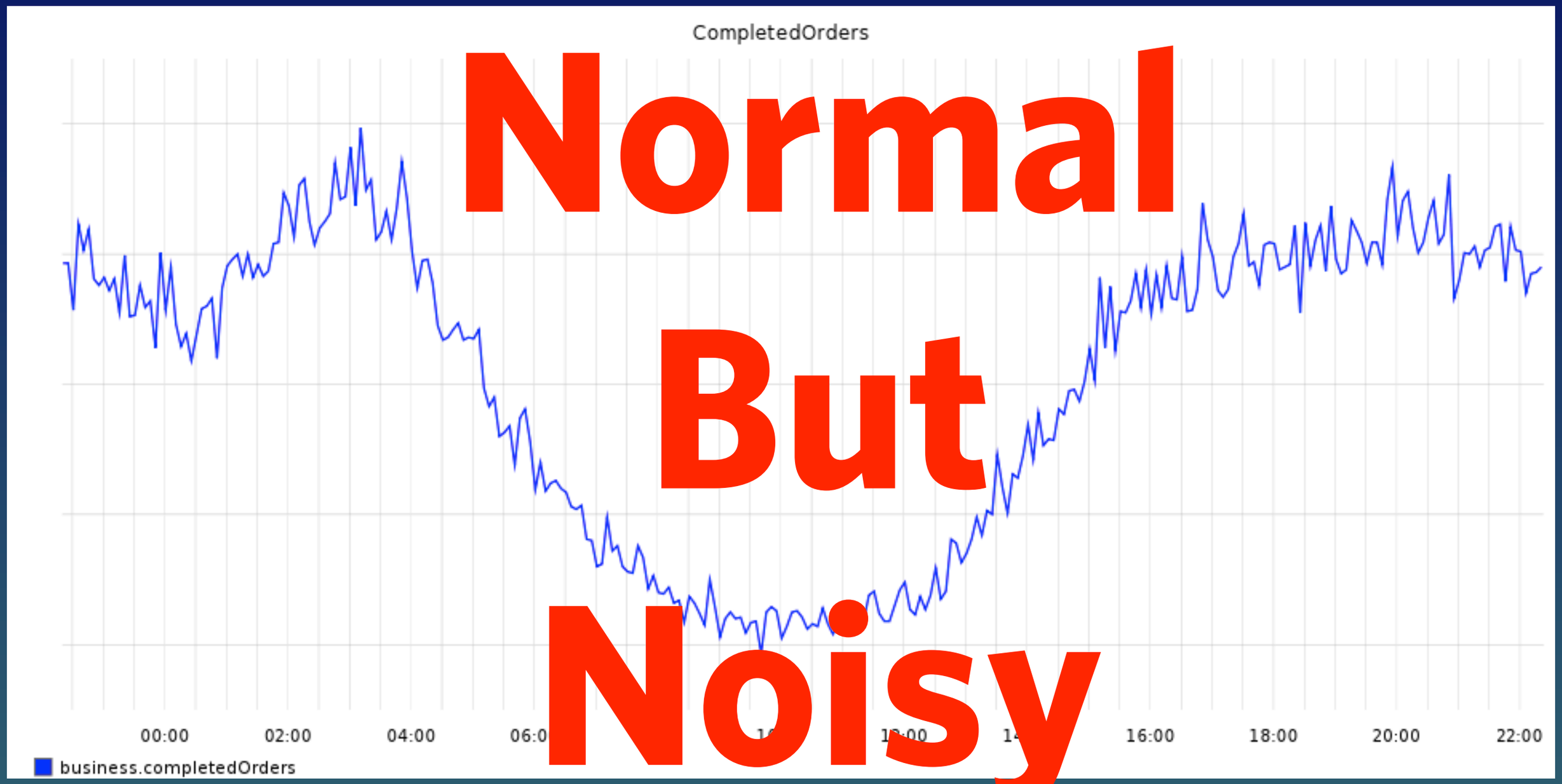
24 hours

Context

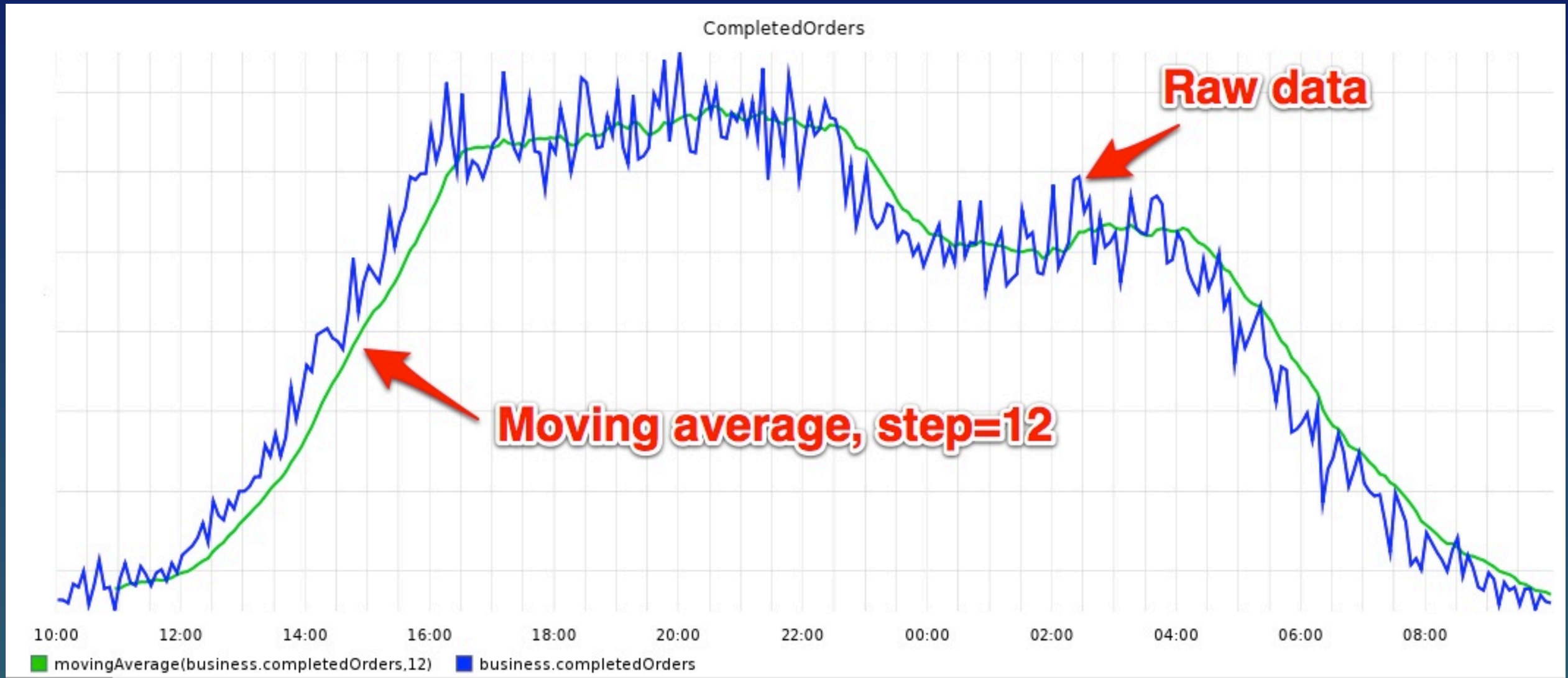


7 days

Context


CompletedOrders
Normal
But
Noisy
00:00
02:00
04:00
16:00
18:00
20:00
22:00
business.completedOrders

Context



Smoothing?

Context

Holt-Winters Exponential Smoothing

Recent points influencing a forecast, exponentially decreasing influence backwards in time.

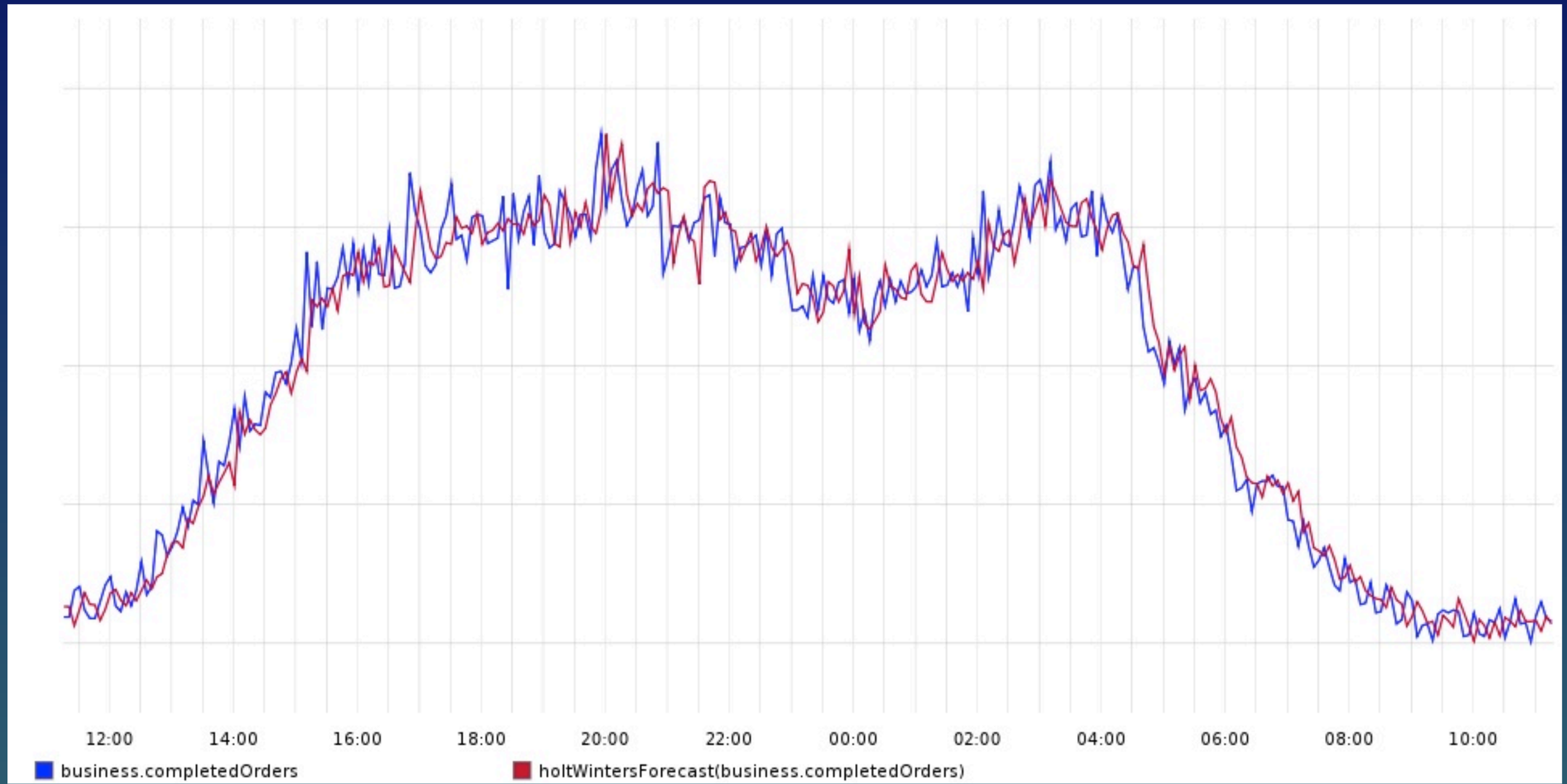
en.wikipedia.org/wiki/Exponential_smoothing

Context

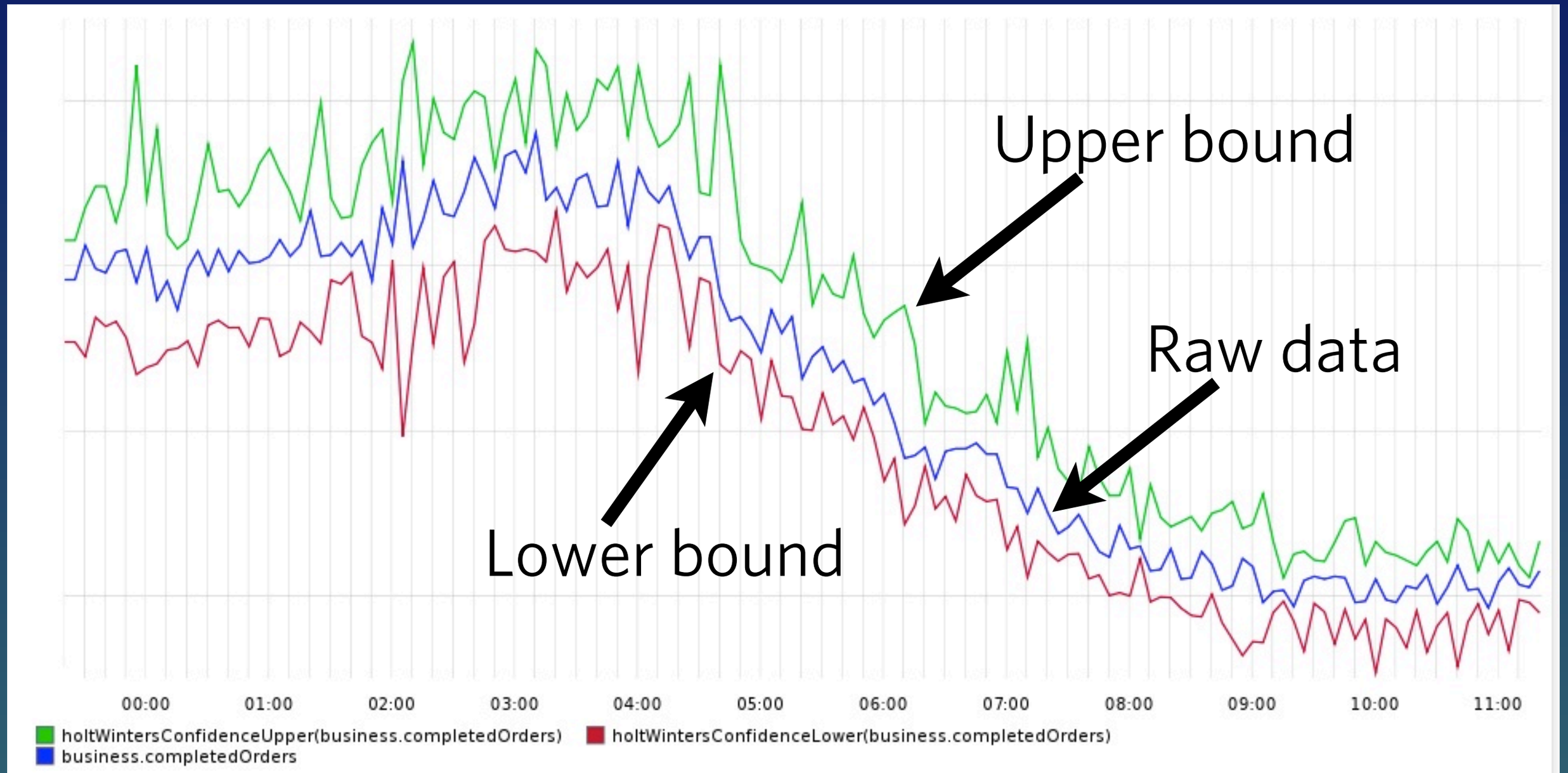
Aberrant Behavior Detection in Time Series for Network Monitoring

http://static.usenix.org/events/lisa00/full_papers/brutlag/brutlag_html/

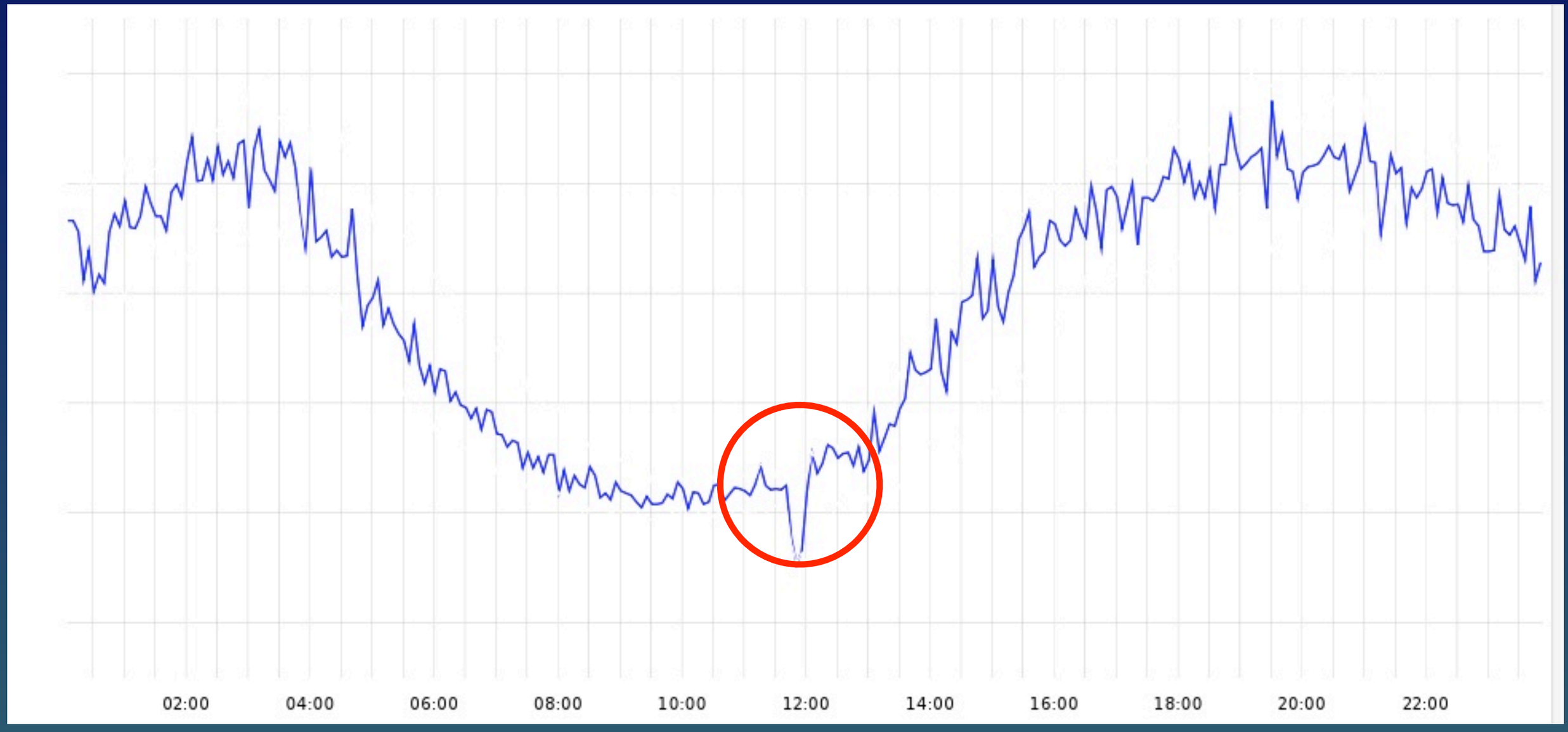
Dynamic Thresholds



Dynamic Thresholds

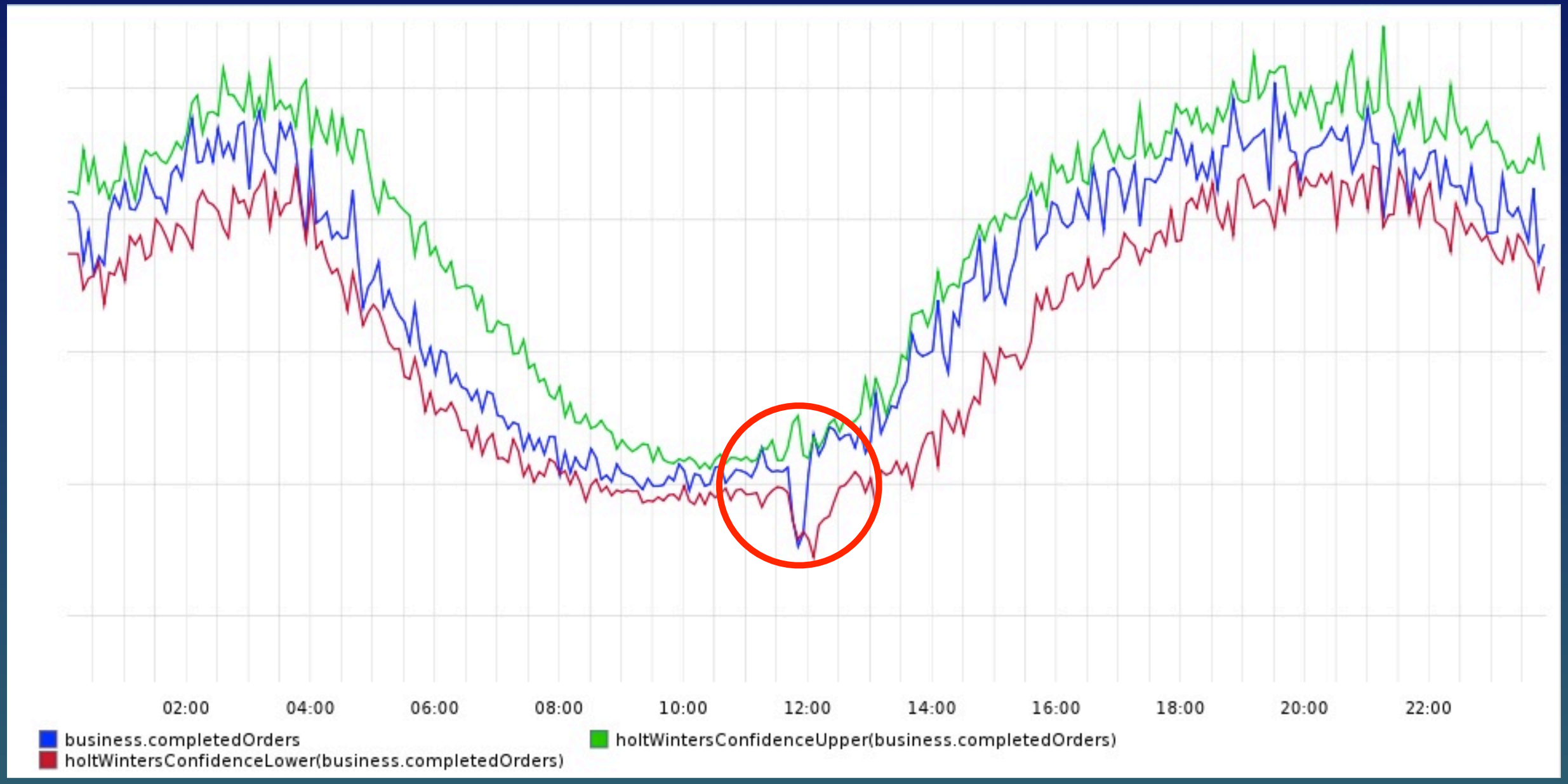


Dynamic Thresholds



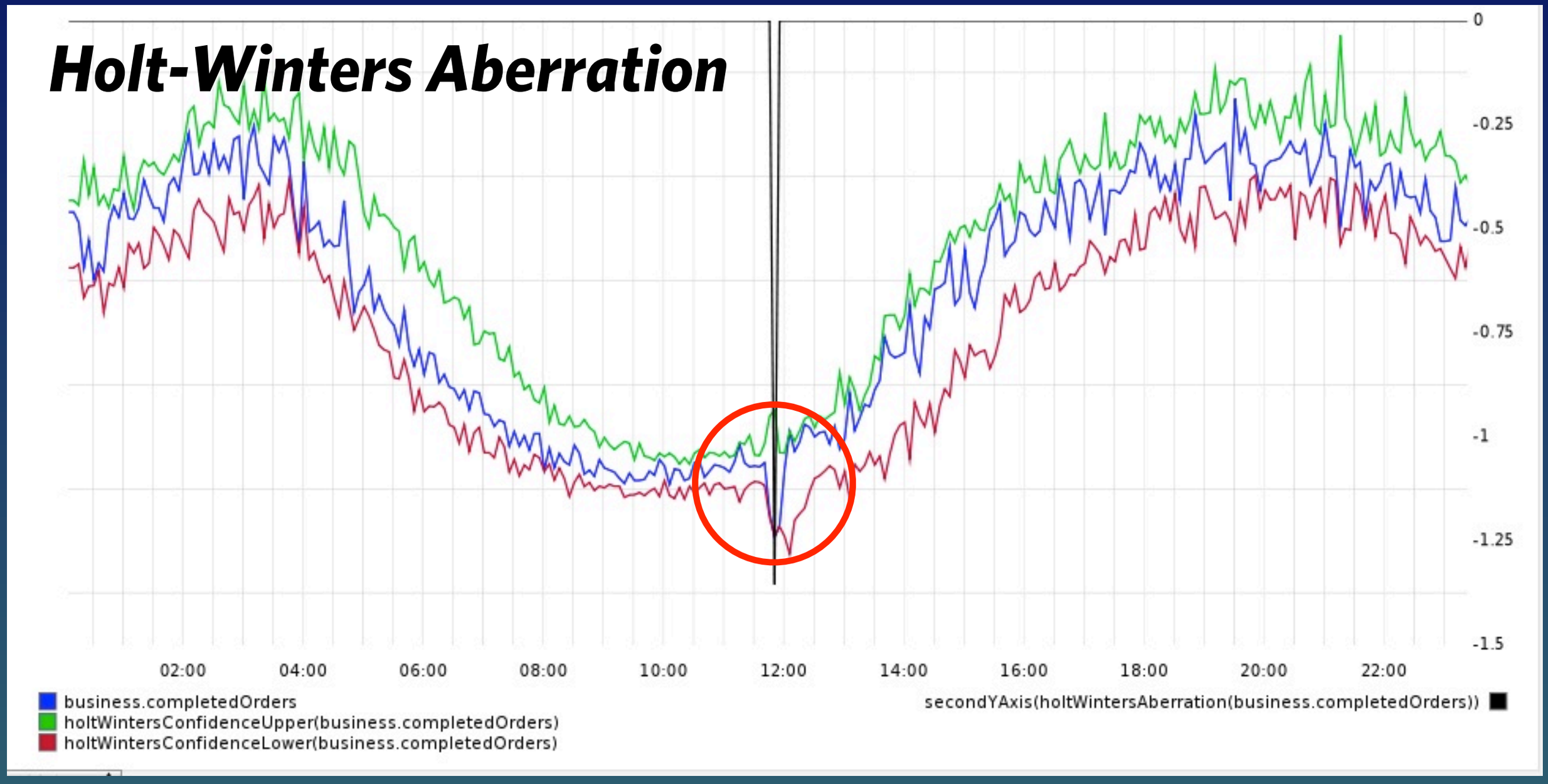
Hrm....

Dynamic Thresholds



Hrm....

Dynamic Thresholds



Ah!

Dynamic Thresholds

Graphite metrics collection w/Holt-Winters abberations

http://graphite.wikidot.com/

Nagios check for Graphite data

https://github.com/etsy/nagios_tools/blob/master/check_graphite_data

Four Cornerstones

Erik Hollnagel

(Anticipation)

Knowing What To Expect

Knowing What To Look For

(Monitoring)

(Response)

Knowing What To Do

Knowing What Has Happened

(Learning)

FAULT

TOLERANCE

Detection of fault X

Triggers corrective action Y

Clean up, report back

(RECOVERY OR MASKING)

Variation Tolerance

Adaptive Systems

Expected Variation

Adaptive Systems

Expected Variation

Adaptive Systems

Expected Variation

New Disturbances Arise

Compensation is Exhausted

Disturbance

Expected Variation

Control

compensation

decompensation

Woods, 2011

New Disturbances Arise

Compensation is Exhausted

Disturbance

Expected Variation

Control

compensation

decompensation

New Disturbances Arise

Compensation is Exhausted

Variation

Disturbance

Expected Variation

Fault

Control

compensation

decompensation

Variations != Faults

Dead
Corrupt
Late
Wrong

Fault Tolerance

Redundancy

Spatial (server, network, process)

Temporal (checkpoint, “rollback”)

Informational (data in N locations)

Fault Tolerance

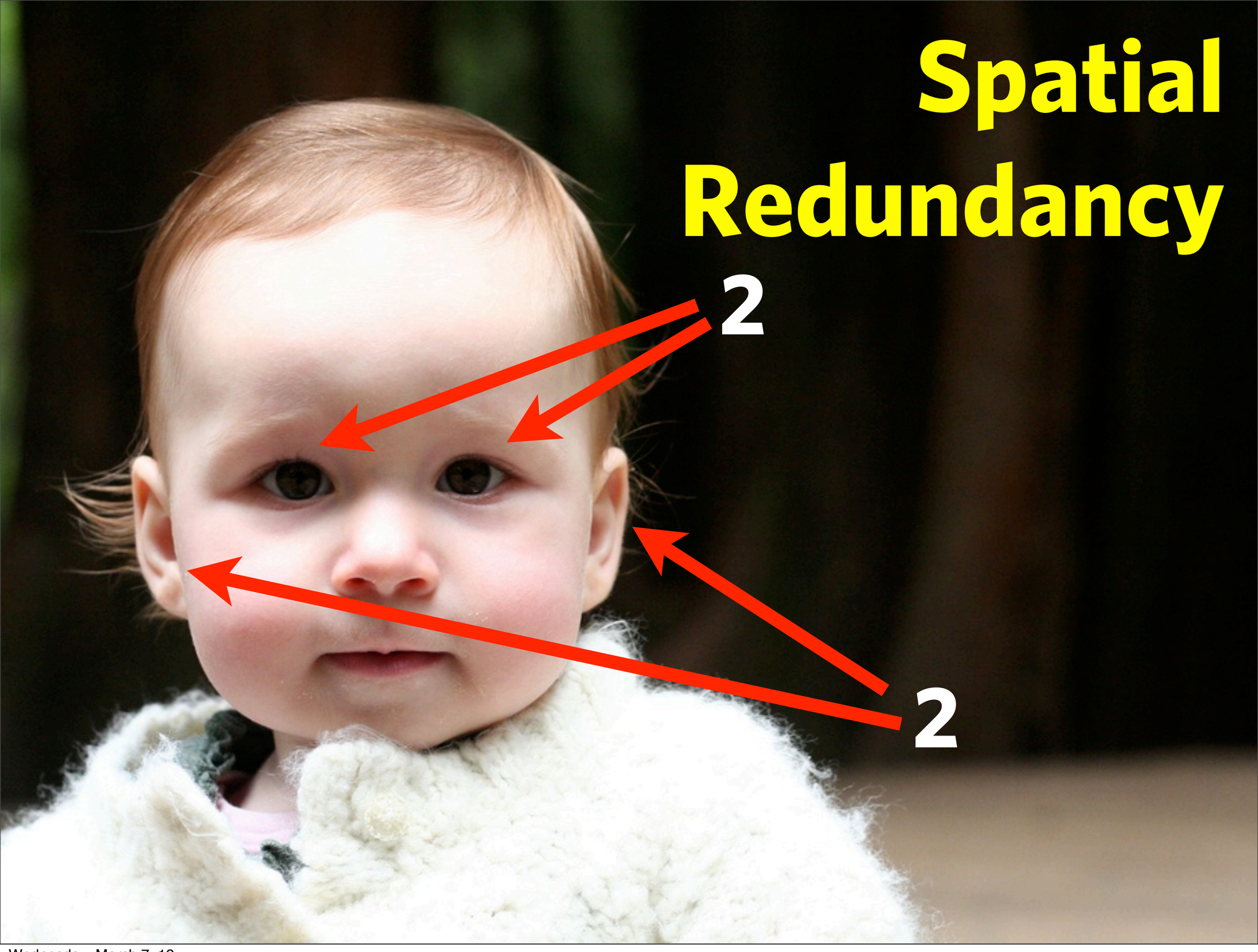
Redundancy

Spatial (server, network, process)

Temporal (checkpoint, “rollback”)

Informational (data in N locations)

Spatial Redundancy



Spatial Redundancy

Active/Active

Spatial Redundancy

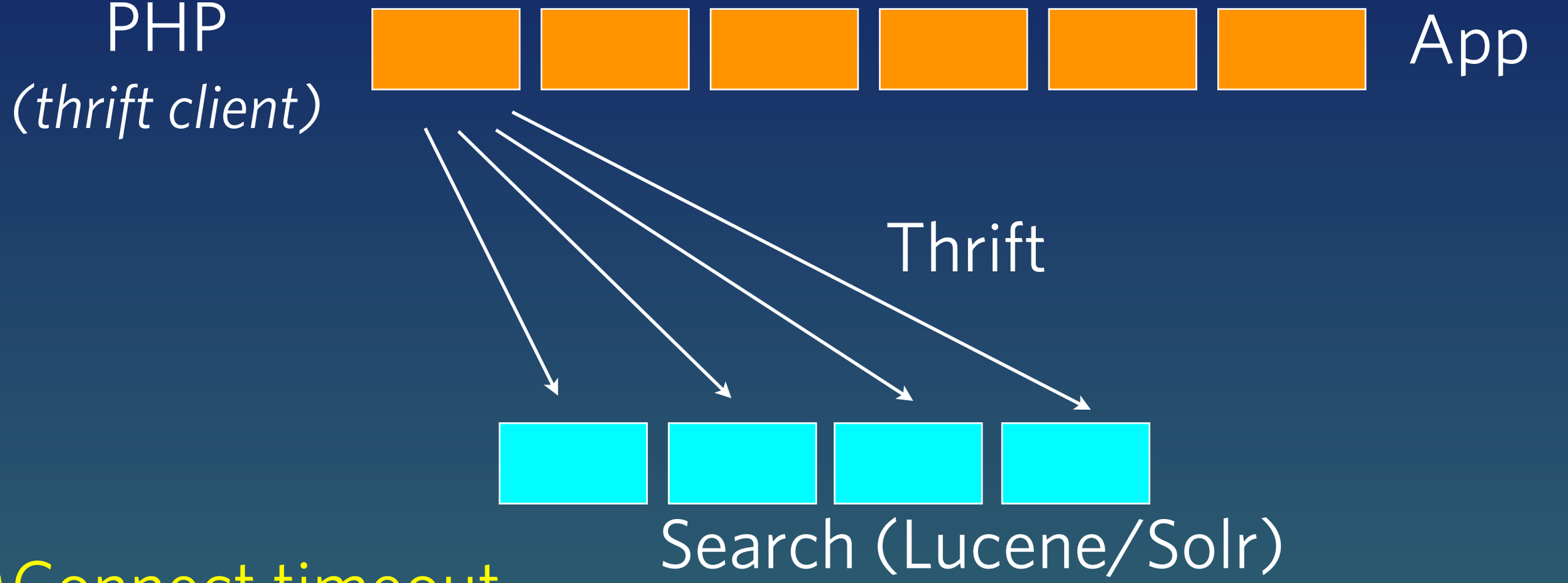
Active/Passive

Spatial Redundancy

Roaming Spare

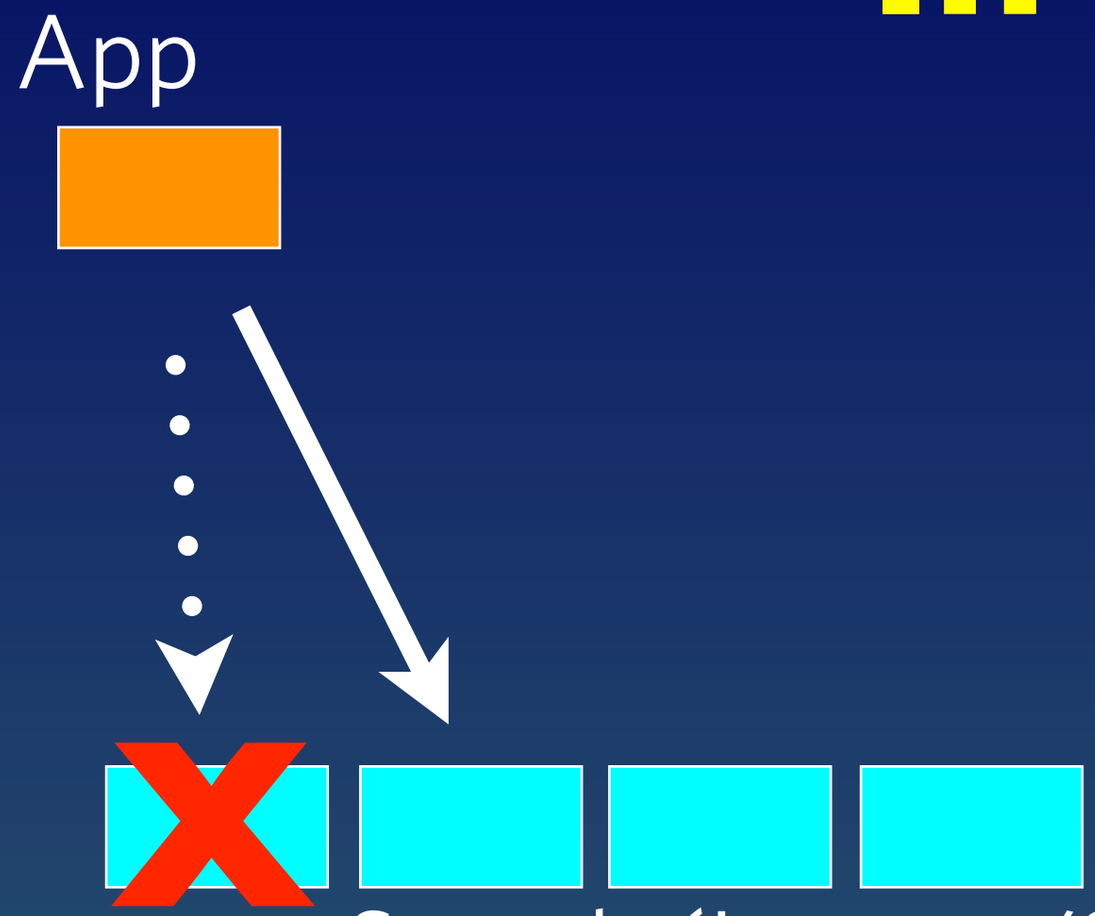
Dedicated Spare

In-Line Fault Tolerance



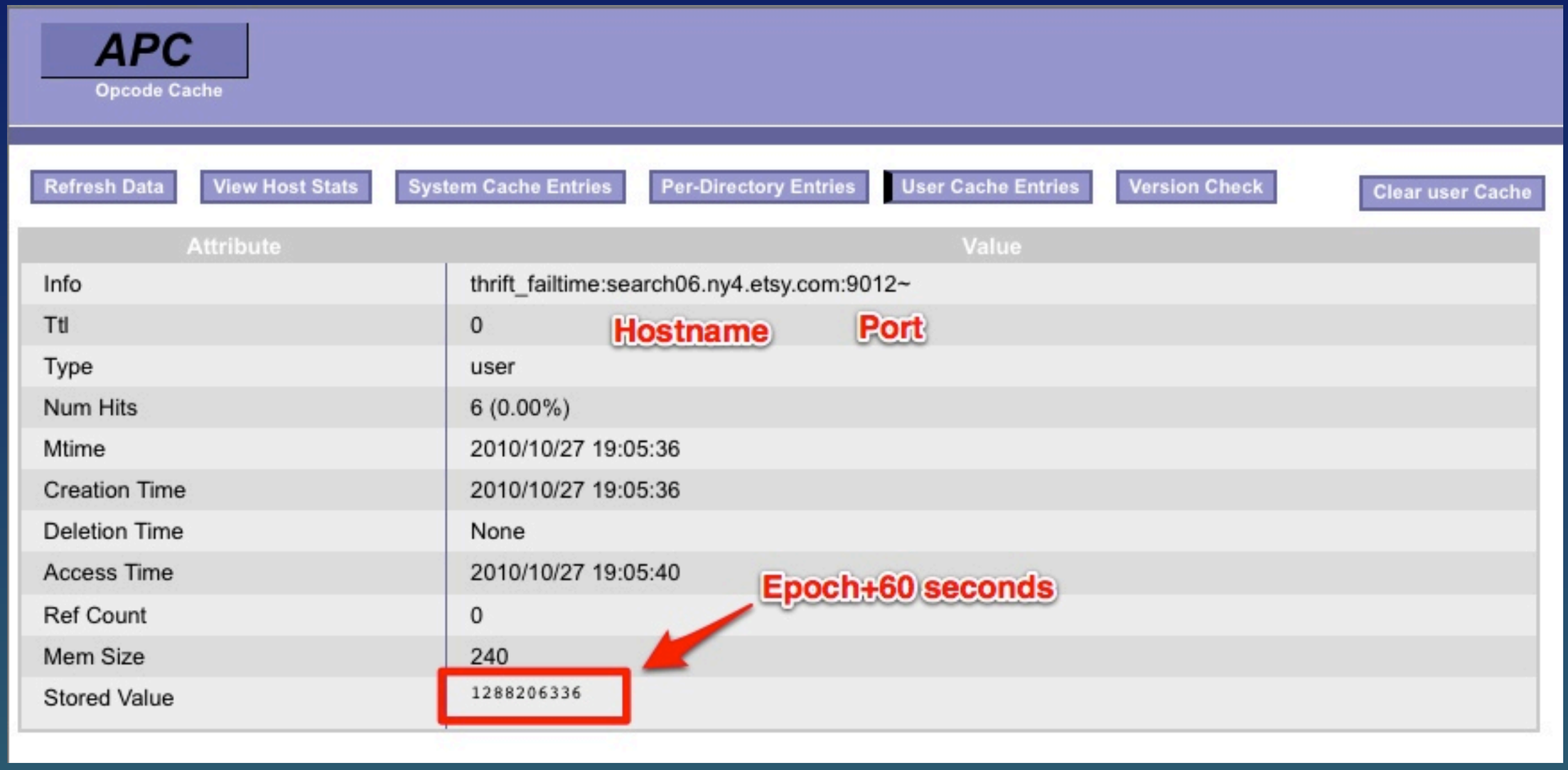
- Connect timeout
- Send timeout
- Receive timeout

In-Line Fault Tolerance



1. App attempts connection, can't
2. Caches APC user object with 60s "TTL" key=server:port
3. Moves to next server in rotation, skipping any found in APC

In-Line Fault Tolerance



http://thrift.apache.org/

/lib/php/src/TSocketPool.php

In-Line Fault Tolerance

Pros:

- Distributed checking and perspective
- Handles transient failures
- Auto-recovery

Cons:

- Onus is on the app for implementation

Fault Tolerance

Nagios Event Handlers

Attempt to recover from specific conditions

Chain together recovery actions

http://nagios.sourceforge.net/docs/3_0/eventhandlers.html

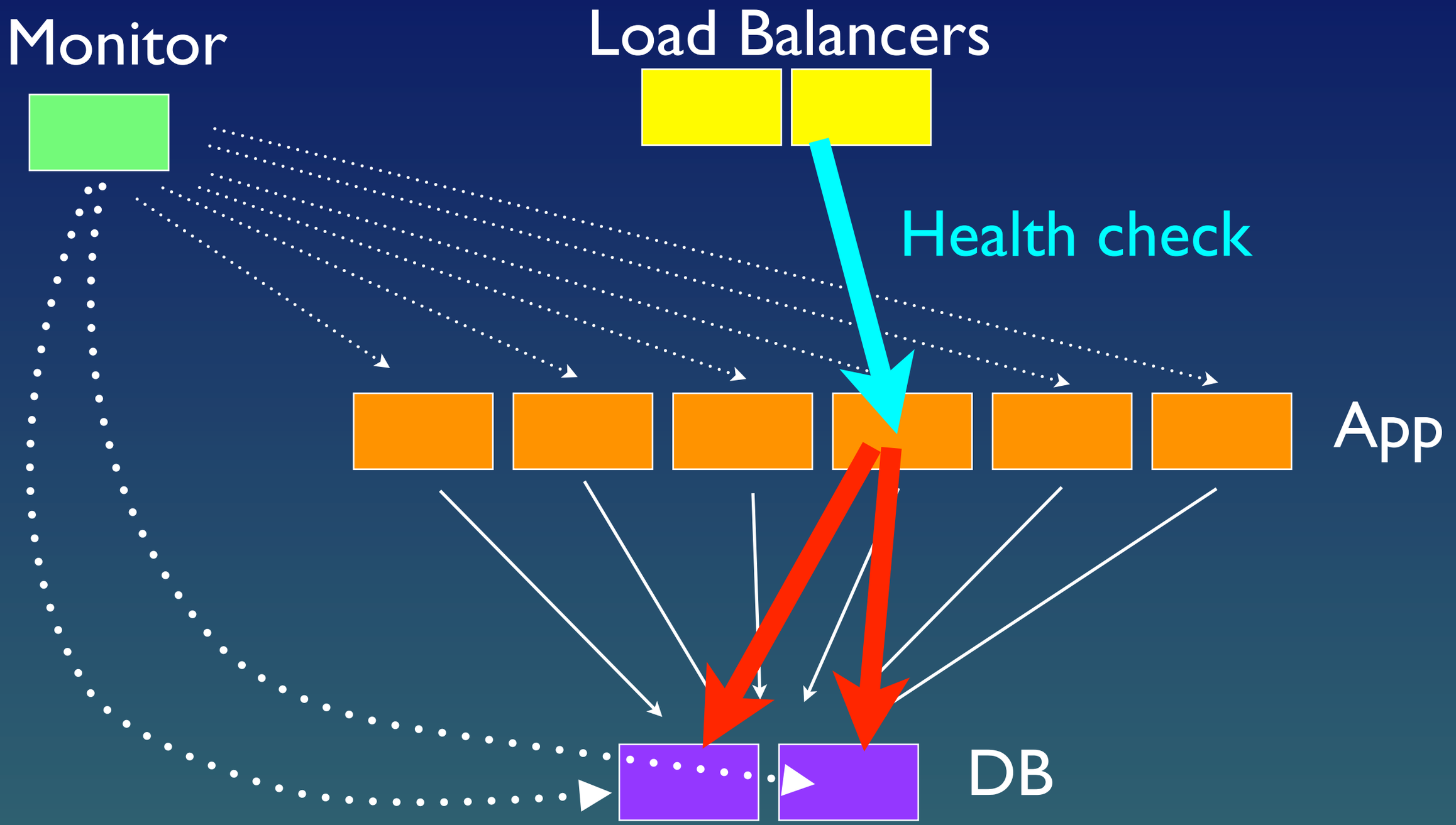
```
If (fault X) then
     HUP process; re-check
       If (OK) then notify+exit
     ELSE
     Hard restart process; re-check
       If (OK) then notify+exit
     ELSE
     Remove from production; notify+exit
```

How many seconds of errors can you tolerate serving?

Fail Closed

When fault is found, and can't be recovered or masked, operations cease to protect the rest of the system from damage.

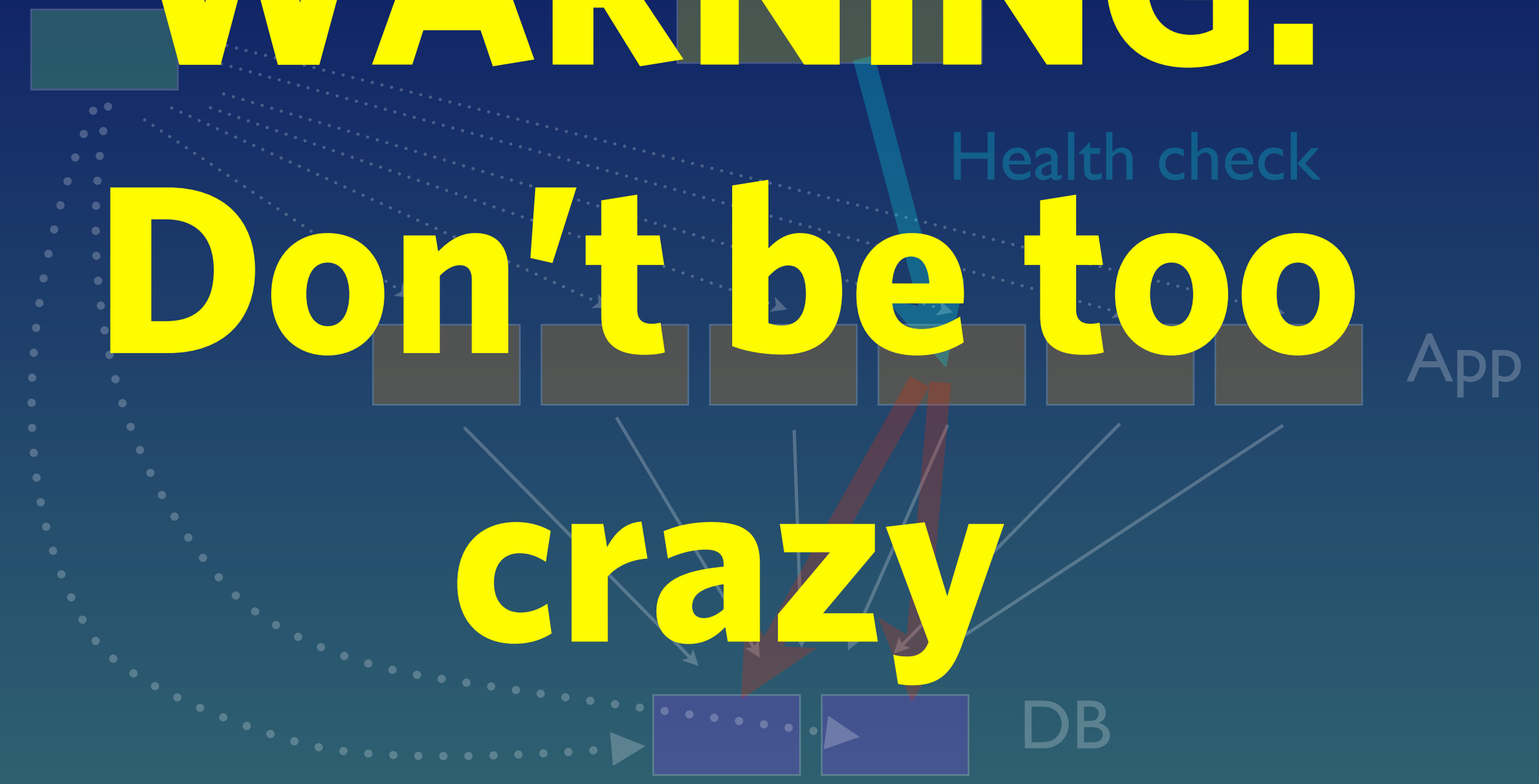
Depth and Dependencies



Depth and Dependencies

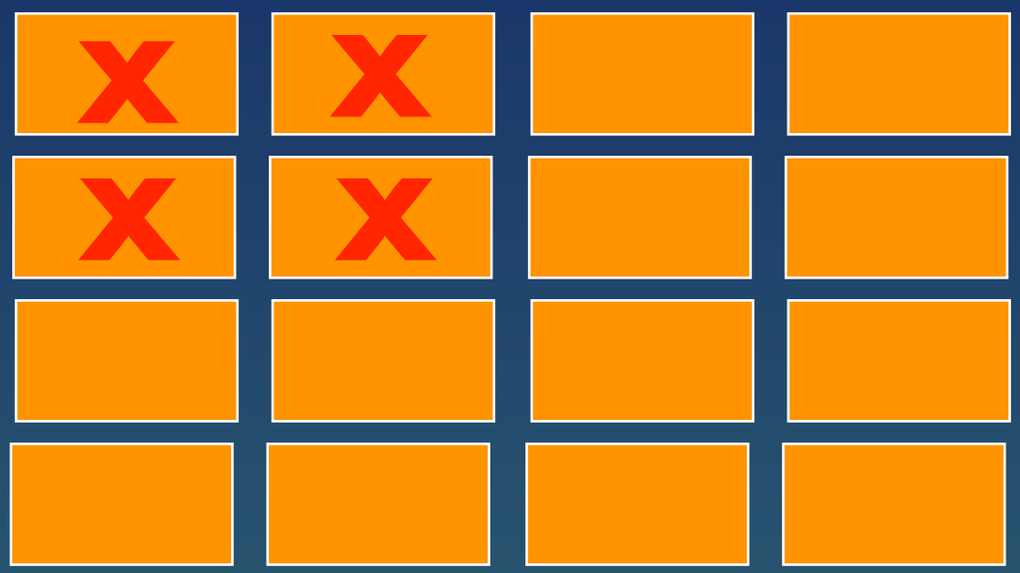
WARNING:

Don't be too crazy



Fail Closed

Aggregate Cluster Checking



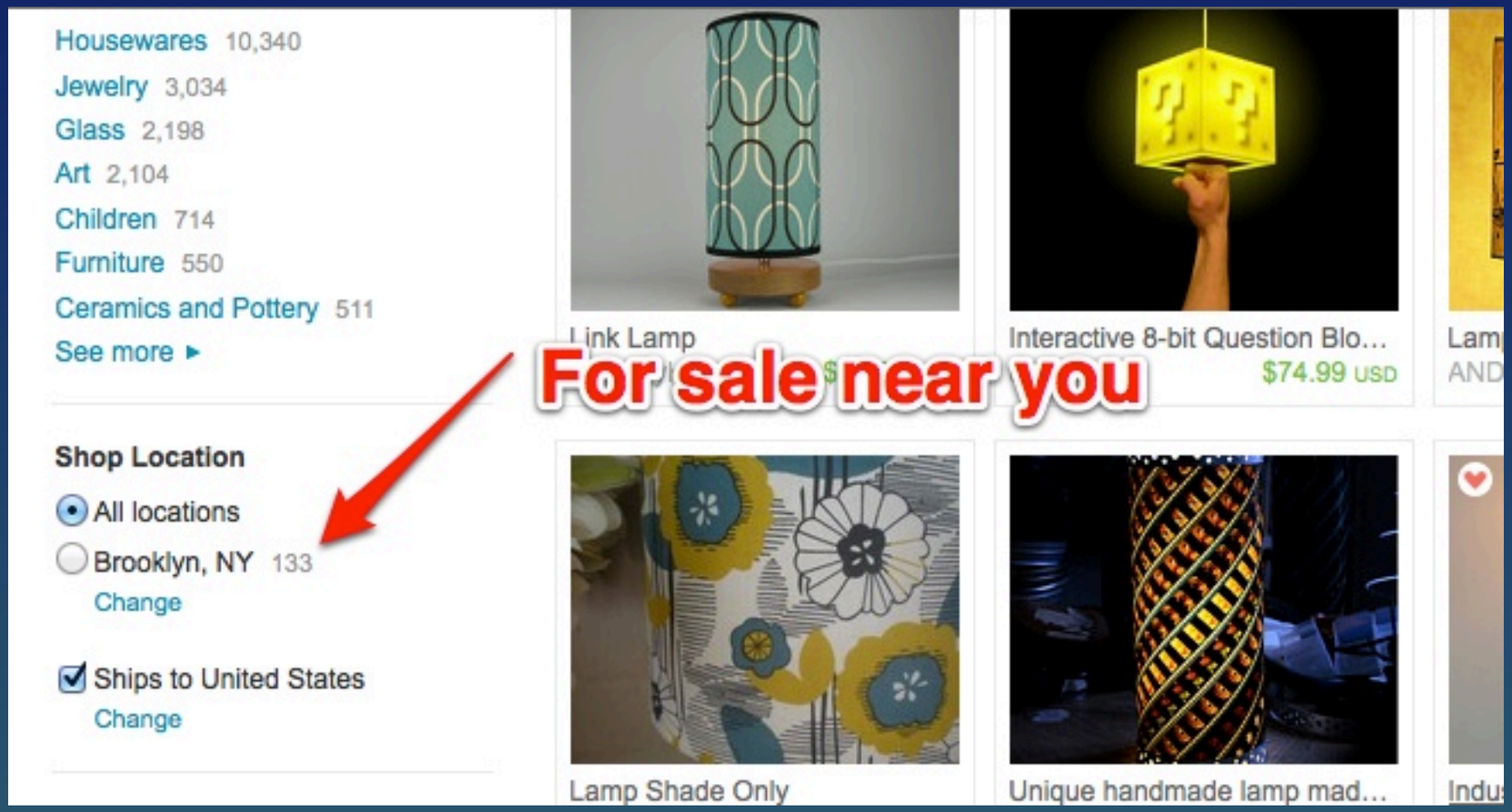
```
If (clusterfail > 25%) then notify+exit

    ELSE OK
```

Fail Open

When a fault happens, and can't be masked or recovered, operations continue without the feature.

Fail Open

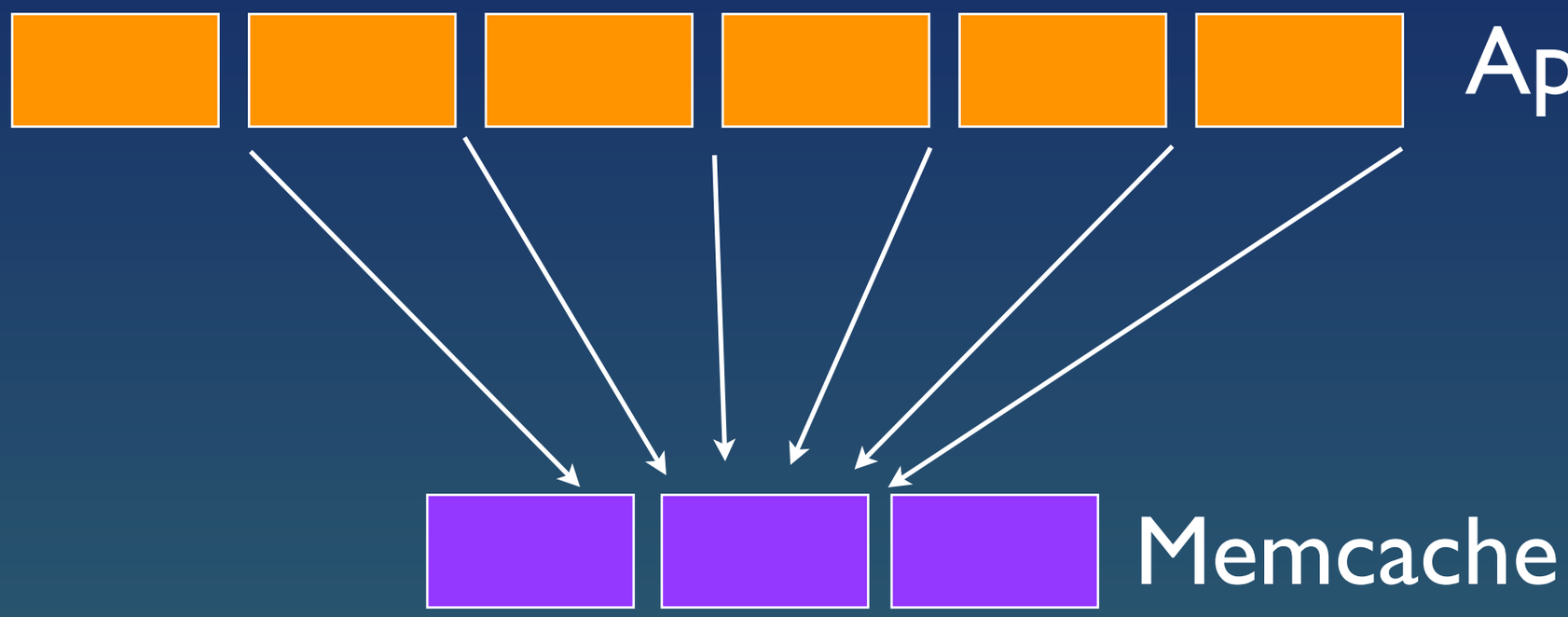
Example 1 at Etsy: Geo Targeting



50ms Internal SLA on guessing location via client IP.
If >50ms, we just don't show local results.

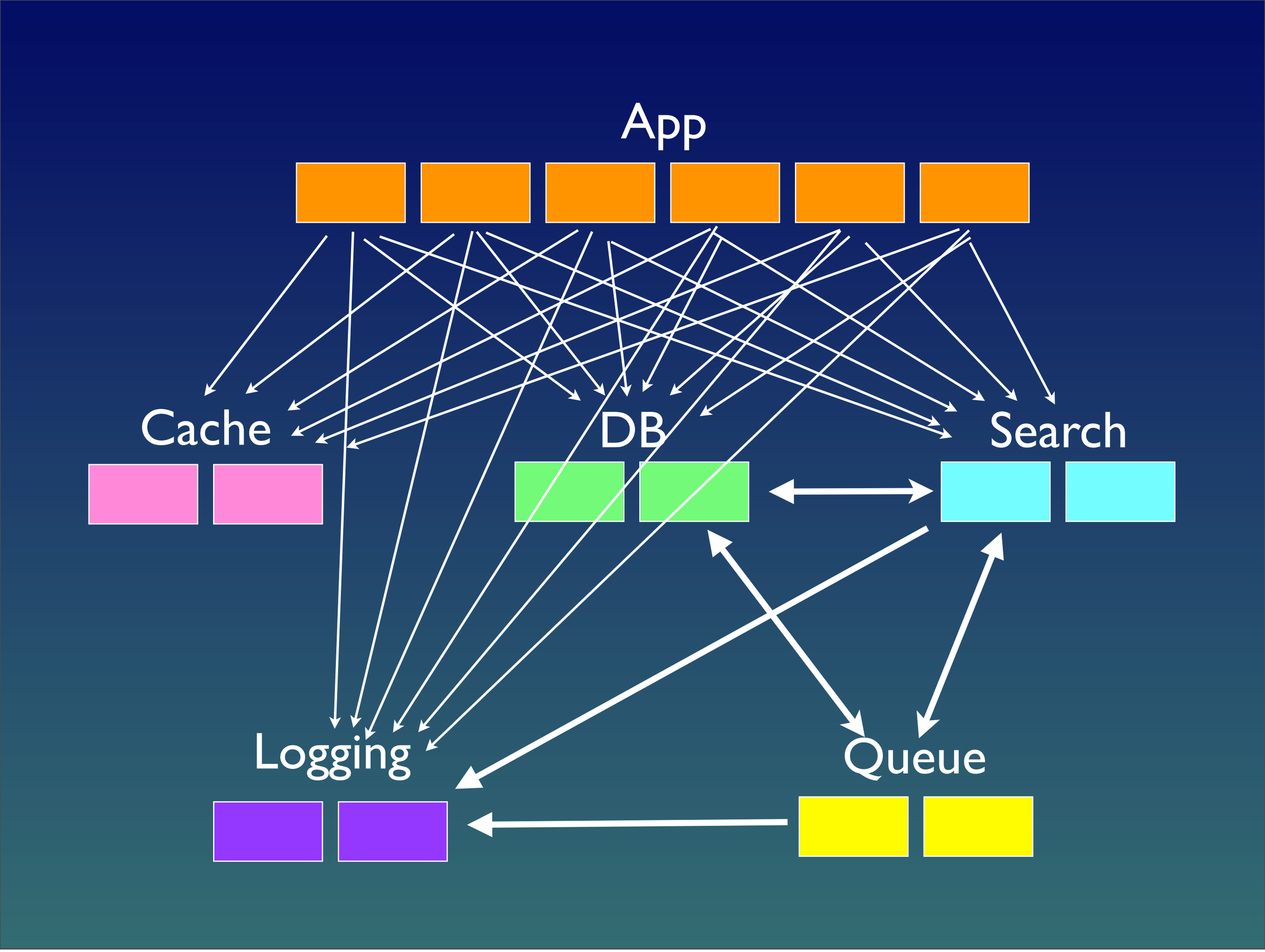
Fail Open

Example 2 at Etsy: Rate Limiting



Internal SLA on incrementing counters+checking totals. If >SLA, we let the action continue, and throw fire-and-forget counter if we can.

SYSTEMIC


App
Cache
DB
Search
Logging
Queue

App

Cache

DB

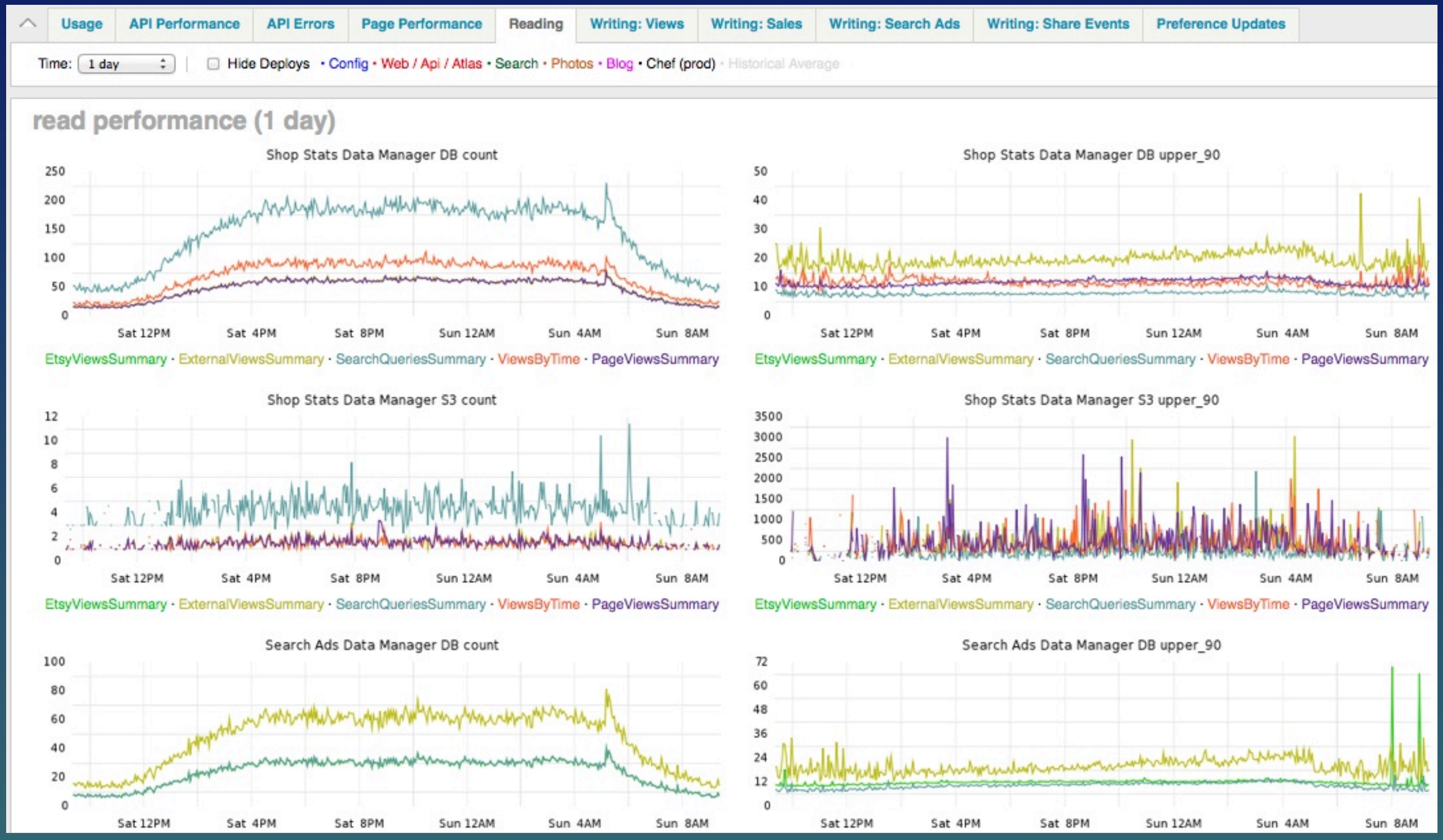
Search

Logging

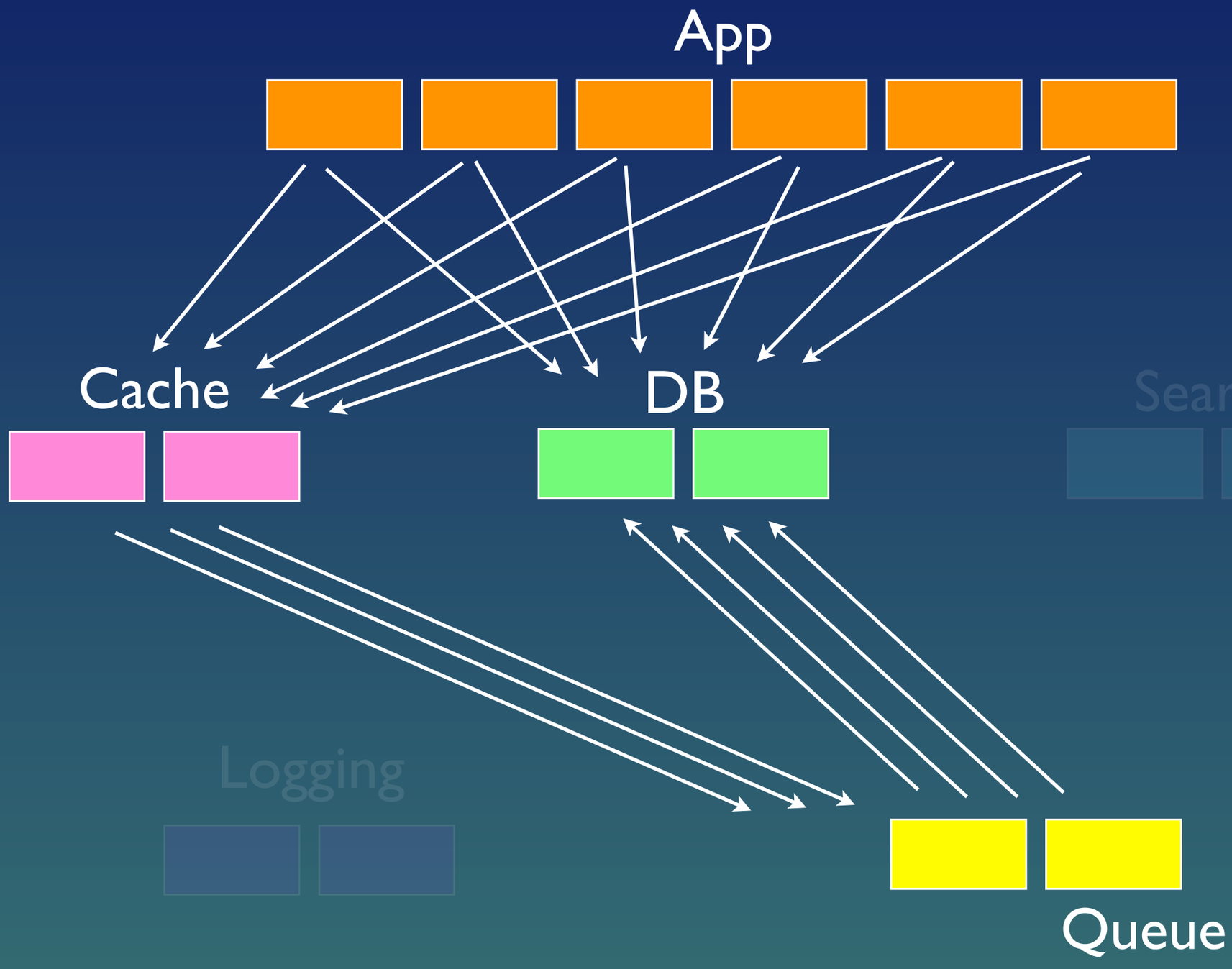
Queue

Functional Resonance

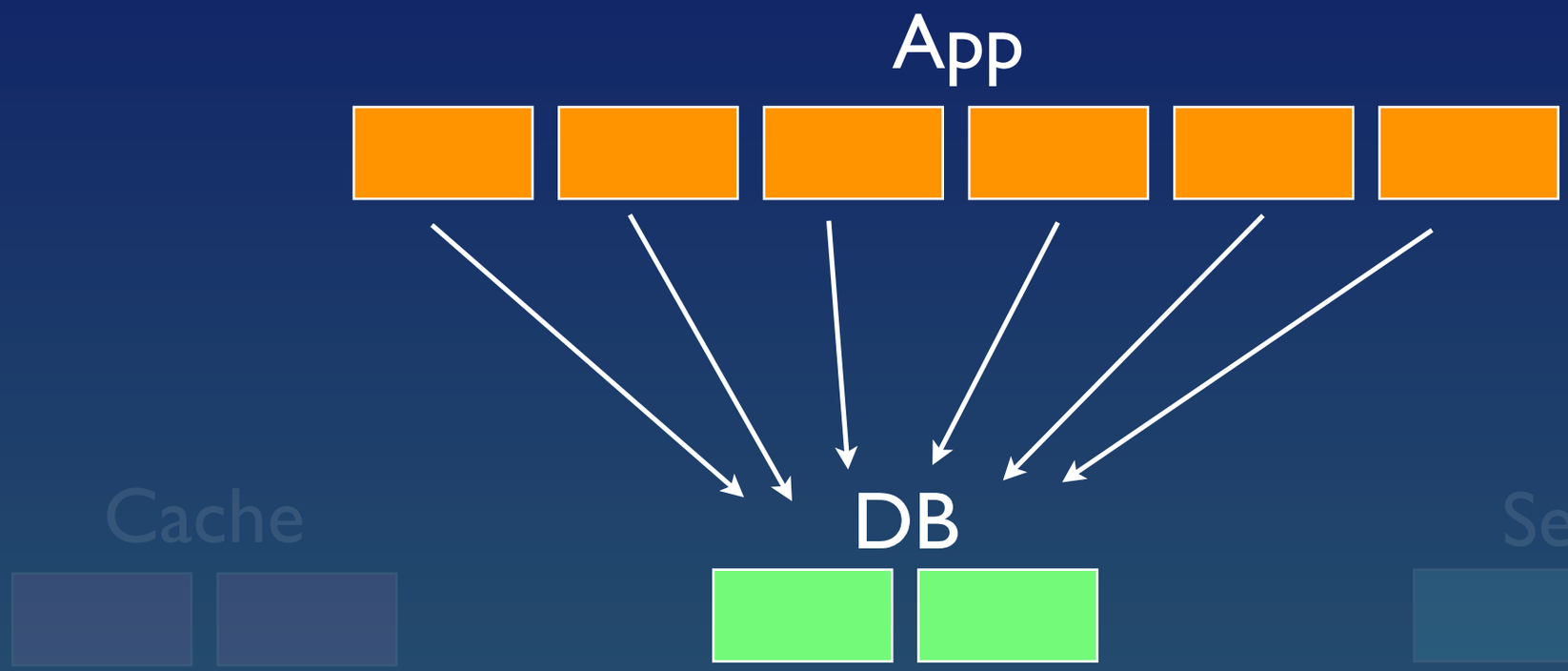
Shop Stats


Usage
API Performance
API Errors
Page Performance
Reading
Writing: Views
Writing: Sales
Writing: Search Ads
Writing: Share Events
Preference Updates
Time: 1 day
Hide Deploys
Config
Web / Api / Atlas
Search
Photos
Blog
Chef (prod)
Historical Average
read performance (1 day)
Shop Stats Data Manager DB count
Shop Stats Data Manager DB upper_90
Shop Stats Data Manager S3 count
Shop Stats Data Manager S3 upper_90
Search Ads Data Manager DB count
Search Ads Data Manager DB upper_90
Sat 12PM
Sat 4PM
Sat 8PM
Sun 12AM
Sun 4AM
Sun 8AM
EtsyViewsSummary · ExternalViewsSummary · SearchQueriesSummary · ViewsByTime · PageViewsSummary

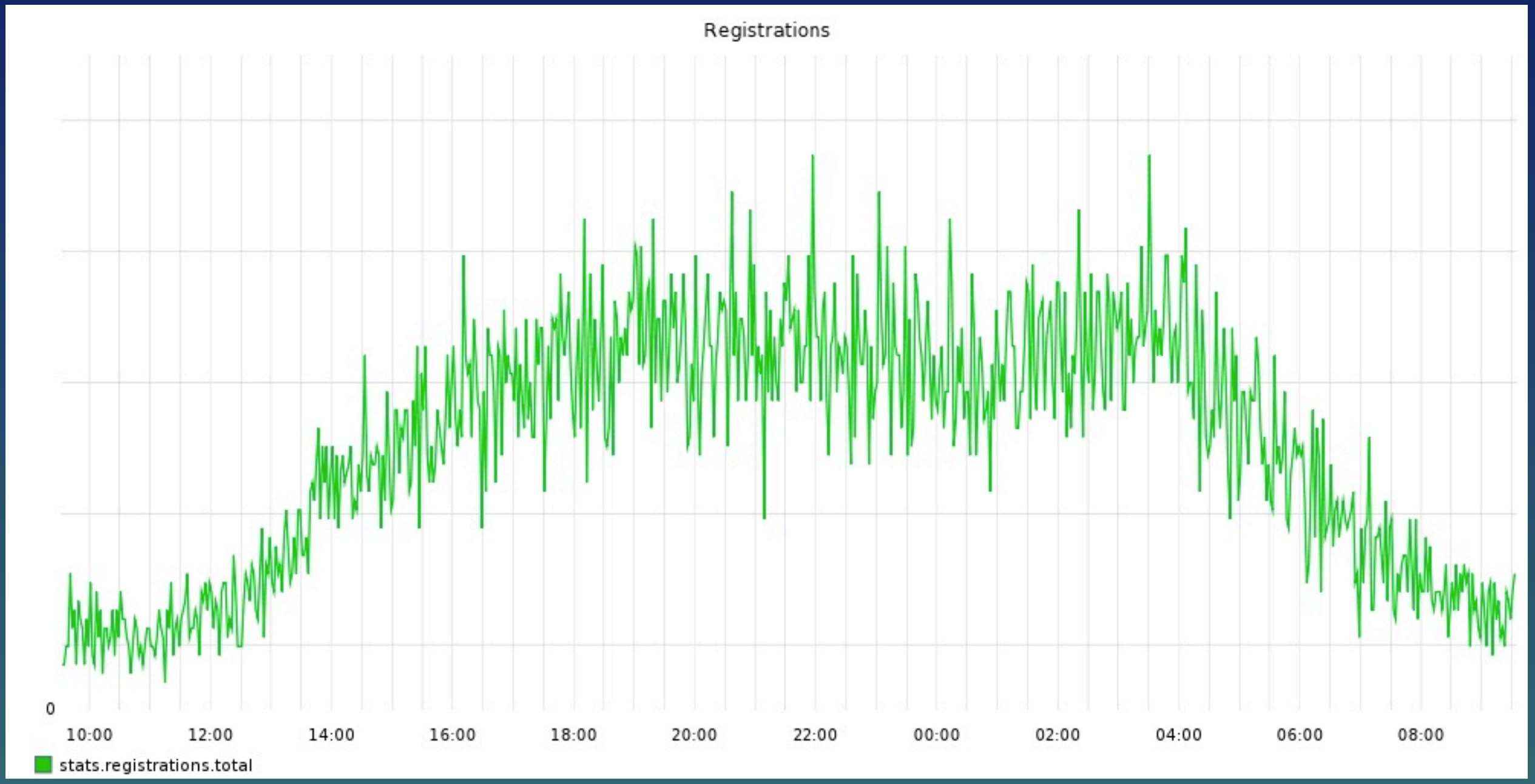
Shop Stats



Registration



Registration



Shop Stats
Logins
Registrations
Checkout
New Listings
Photos

Search
API
Rate limiting
Data Analysis
Search
A/B analysis

Page performance
Search Ads
Editorial content
Email systems
Feedback
Messaging/Convos

Activity Feeds
Circles
Shipping
Mobile
Internationalization
Testing
Fraud

Systemic

Application/Functionality Health

Componential/Resource Health

Four Cornerstones

Erik Hollnagel

(Anticipation)

Knowing What To Expect

Knowing What To Look For

(Monitoring)

(Response)

Knowing What To Do

Knowing What Has Happened

(Learning)

Anticipation

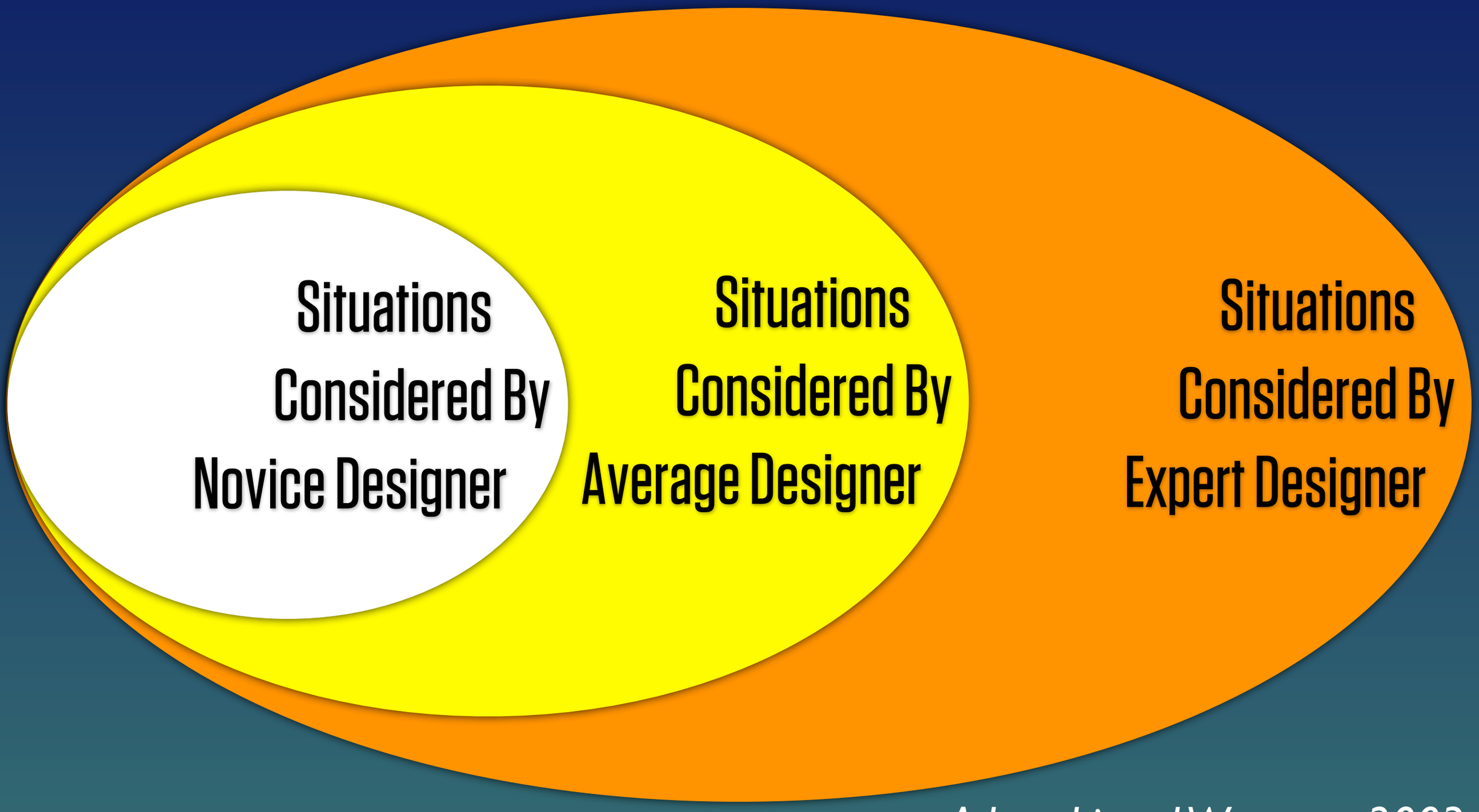
- During design of architecture
- During choice of technologies
- During design of monitoring and metrics

TRADE-OFFS

“What could possibly go wrong?”

REQUISITE IMAGINATION

Possible Foreseeable Situations



Adamski and Westrum, 2003

Anticipation

Failure Mode Effects Analysis (FMEA)

http://en.wikipedia.org/wiki/Failure_mode_and_effects_analysis

Failure Mode Effects and Criticality Analysis (FMECA)

http://en.wikipedia.org/wiki/Failure_mode,_effects,_and_criticality_analysis

Architectural reviews

Go or No-Go meetings

“Game Day” exercises

Anticipation

Servers
Networks
Software

Applications
Monitoring
Metrics
Traffic

PEOPLE

(Anticipation)

(Response)

Knowing What To Expect

Knowing What To Look For

Knowing What To Do

Knowing What Has Happened

(Monitoring)

(Learning)

THE END