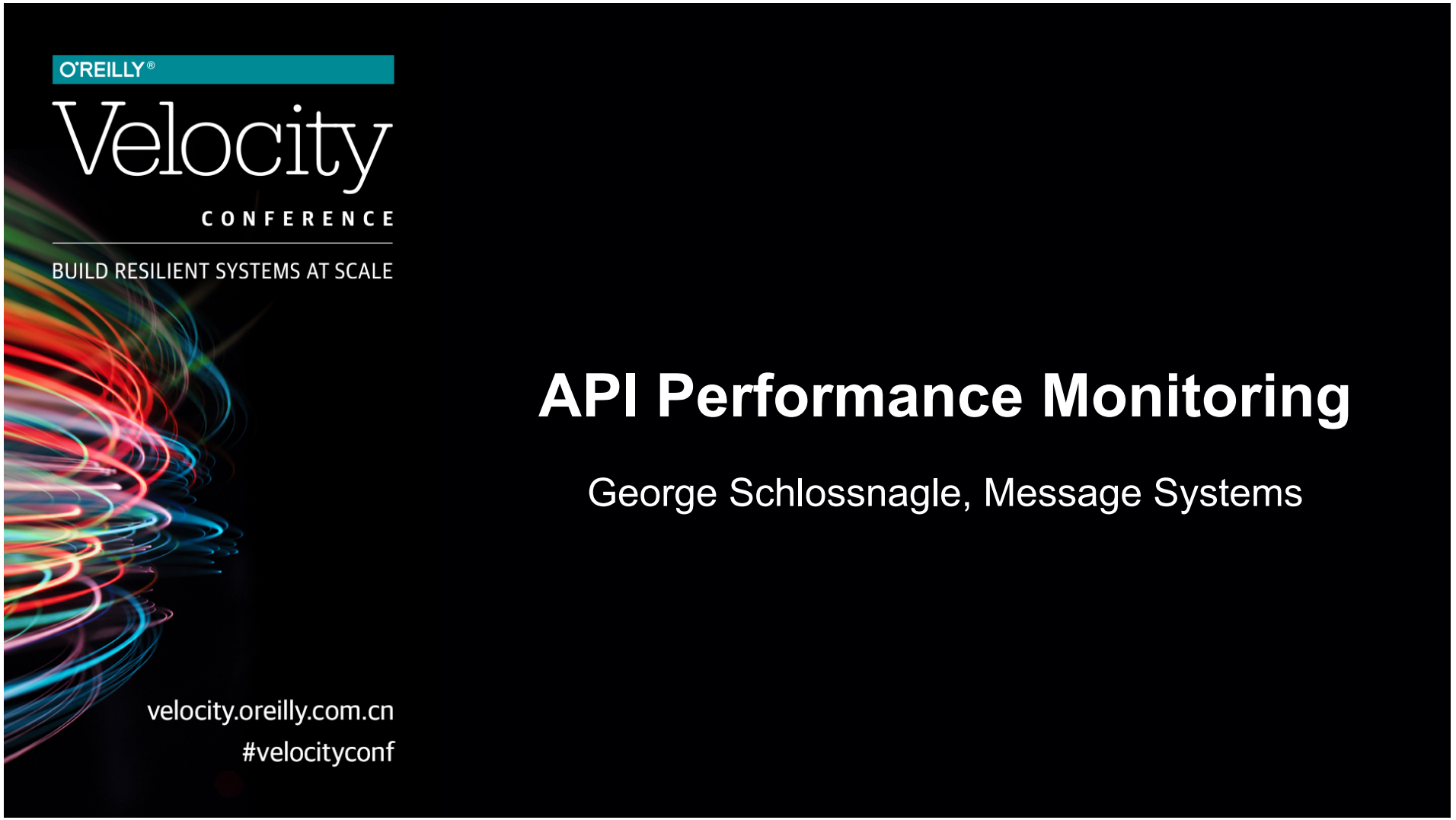


API Performance Monitoring

George Schlossnagle, Message Systems

If we have data, let's look at the data. If all we have are opinions, lets go with mine.

-Jim Barksdale

About Message Systems

- World's largest provider of email messaging infrastructure
- Traditionally delivered as on-premise software, moved into the cloud recently
  - ~ 30B message per day on-premise, ~250M messages per day in the cloud
- Platform fully accessible through multiple developer-focused APIs

Some of our Customers

Our Agenda

- General goals for monitoring
- Active monitoring
- Passive monitoring
- Some other topics

A note about tools

- This is a talk about monitoring strategy and general techniques, not about specific tools
- Most of what we use is built around Circonus with a large custom code base of agents and infrastructure
- Plenty of alternative options:
  - Nagios
  - Graphite
  - Reconnoiter
- Depending on your platform, investing some time in R or NumPY may be useful

General Qualities We Want in Monitoring

- Real Time Analysis
- Trending/Analytics
- Anomaly Detection
- Alerting
- Ease of Deployment and Provisioning

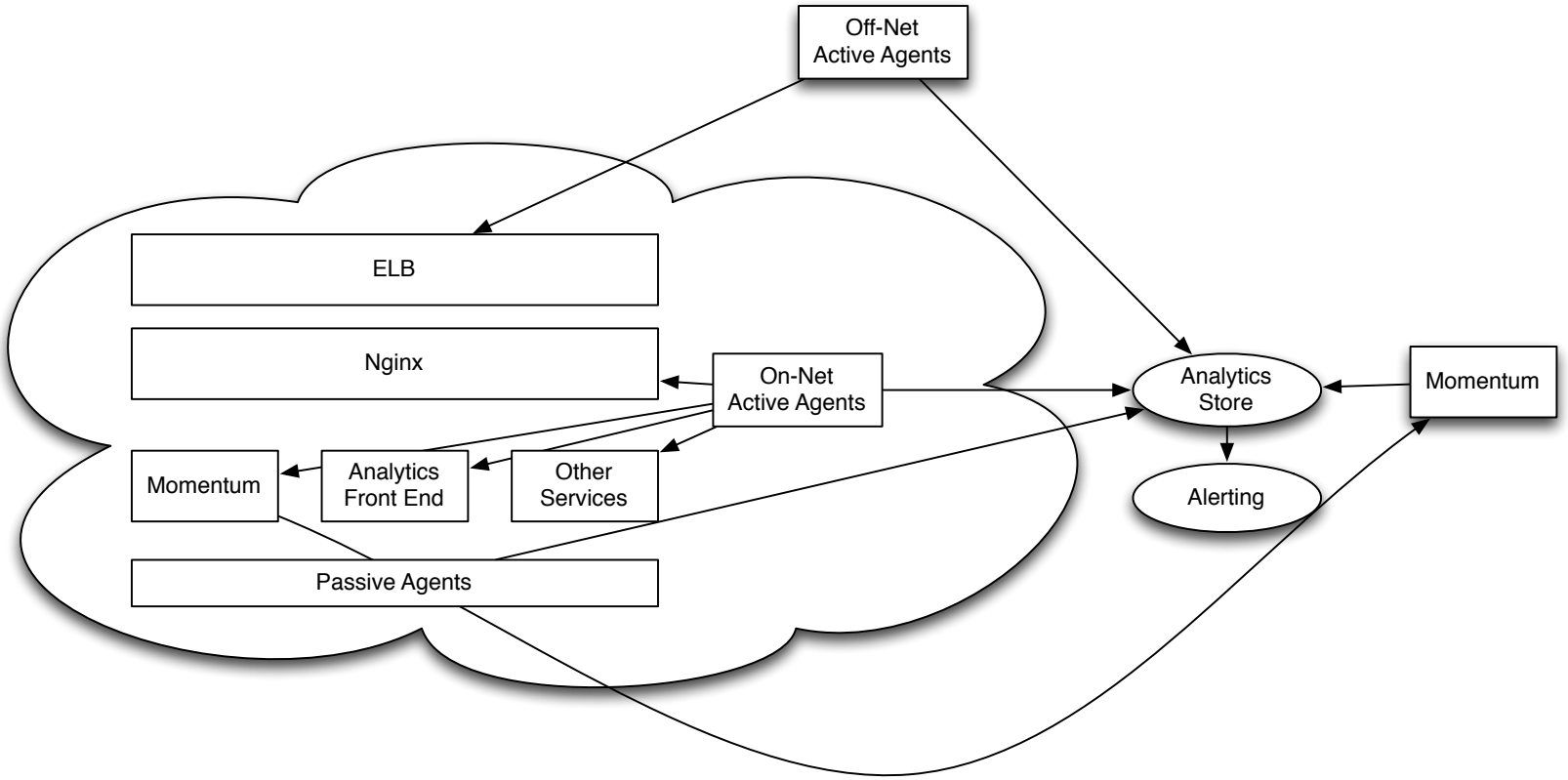
The Difference Between Monitoring and Alerting

- Monitoring
  - Collecting telemetry data that may be interesting
  - Good for analytics/trending
  - Good for contextualizing issues
  - Good for understanding typical runtime behavior
  - Can drive and inform alerting strategy
- Alerting
  - Should only be used for actionable events
  - Beware of 'boy who cried wolf'
  - Thresholds should be a compromise between reality and your ideals

Two Types of Monitoring

- Active Monitoring
  - Triggered tests
  - Constrained by time and ingenuity
- Passive Monitoring
  - Colleting telemetry off of 'real' traffic
  - Less depth, but real population data

A Simplified View of our Monitoring Infrastructure



Simple Active Monitoring

- Direct probing of APIs
- Known payloads/expected responses
- Pros:
  - You control the cadence of the check
  - You can target areas you're interested in
  - You can trigger from wherever you want (internal to network, external)
  - Since you control the call, you can collect any telemetry data you choose
- Cons
  - Not necessarily representative of all user experiences

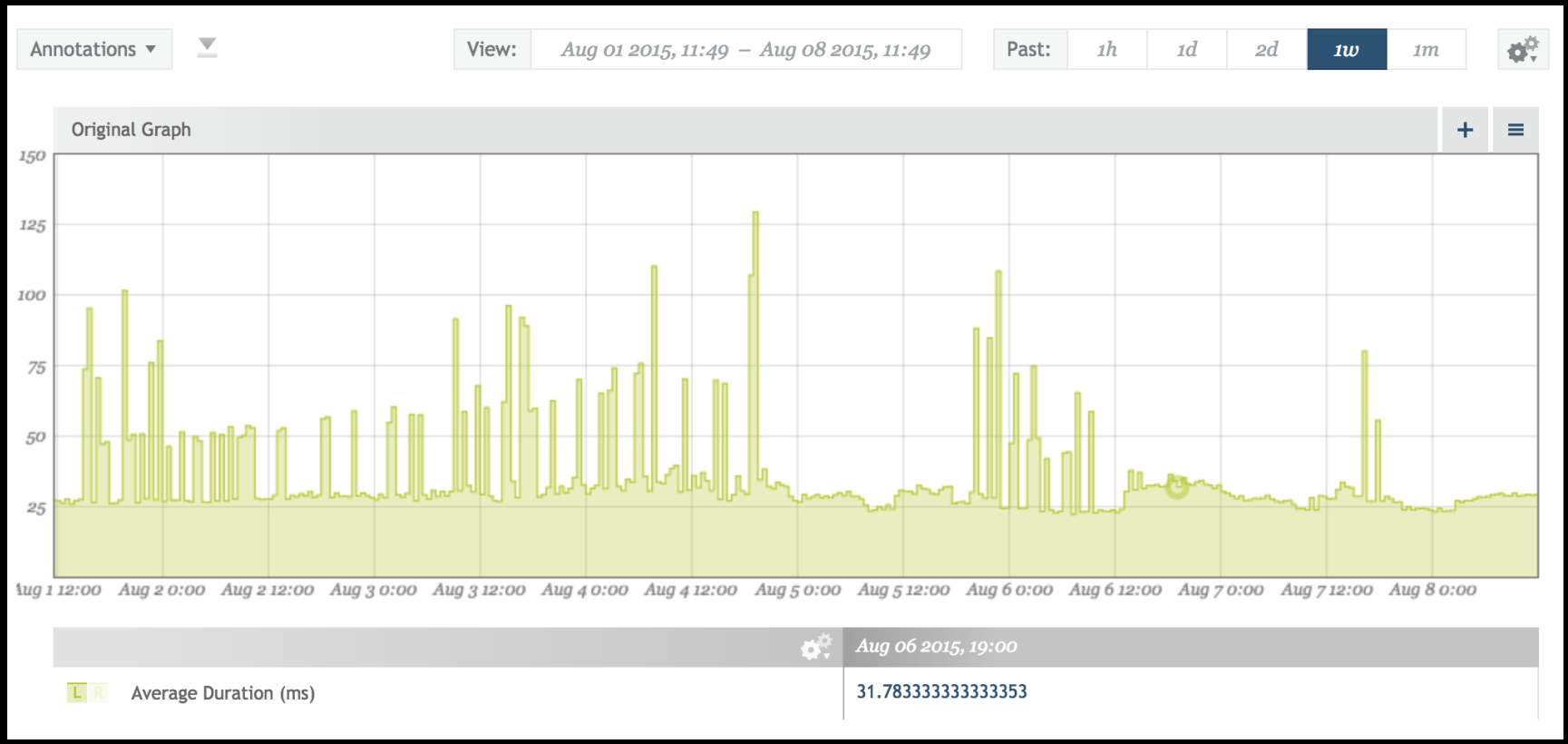
Rich Telemetry Data from Active Monitors

- Typical Information to Return
  - DNS resolution time
  - Connection time
  - Time till initial response
  - Time till complete response
  - Payload information
- Custom application data
  - You can enhace your APIs to return custom internal information about internal performance telemetry
  - Data store access times, template rendering times, etc.

Frequency and Targeting of Active Checks

- Internal to our network
  - Every API endpoint
  - Every major variation
  - Full CRUD checks, where possible
  - Every host
  - Every minute
- External to our network
  - All of the above
  - Exercise full customer-facing authentication
  - Execute from multiple networks

Simple Probe of Transmissions API



Active Monitoring of Complex Workflows

- Direct probing of APIs and validation of expected results
- Handling actions with consequencese
- Good for evaluating whether the *service* works, vs. whether the API works
- Often complex and requires more ellaborate testing infrastructure

Active Monitoring of Complex Workflows

- Pros:
  - Provides holistic view of service offering
  - Validates that the servce itself is actually working, not just the API call
- Cons
  - Complicated to implement, many moving parts
  - Tests only a specific workflow
  - If service offering involves external components, may be reliant on things outside your control

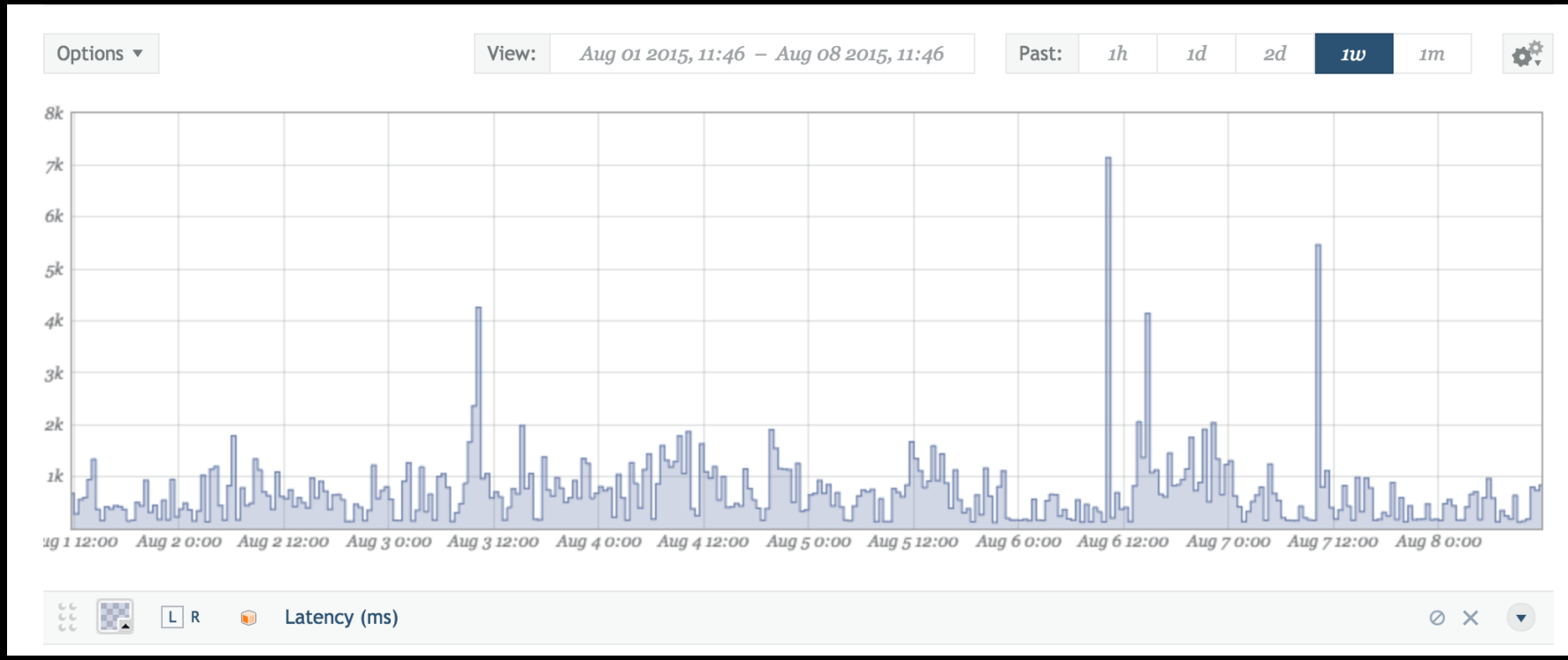
An Example: Suppression API

1. Create a random address
2. Make a suppression API creation call and check its return
3. Make a GET call to validate it created
4. Attempt to send a mail to the suppressed address and validate it gets suppressed
5. Make a suppression API deletion call and check its return
6. Make a GET call to validate it no longer exists

A Second Example: End-to-end Transmission Testing

1. Use message content that will contain a known link that we can expect to be wrapped
2. Make a transmissions API call to a known mailbox were we can receive and process messages.
3. Tag the message with metadata so that when we receive it we can:
    1. Determine eaxactly how long it took to get to us
    2. Validate that the crypto signature was applied correctly
    3. Validate that the link tracking was performed and actually resolves correctly
    4. Validate that the email arrived via the correct IP

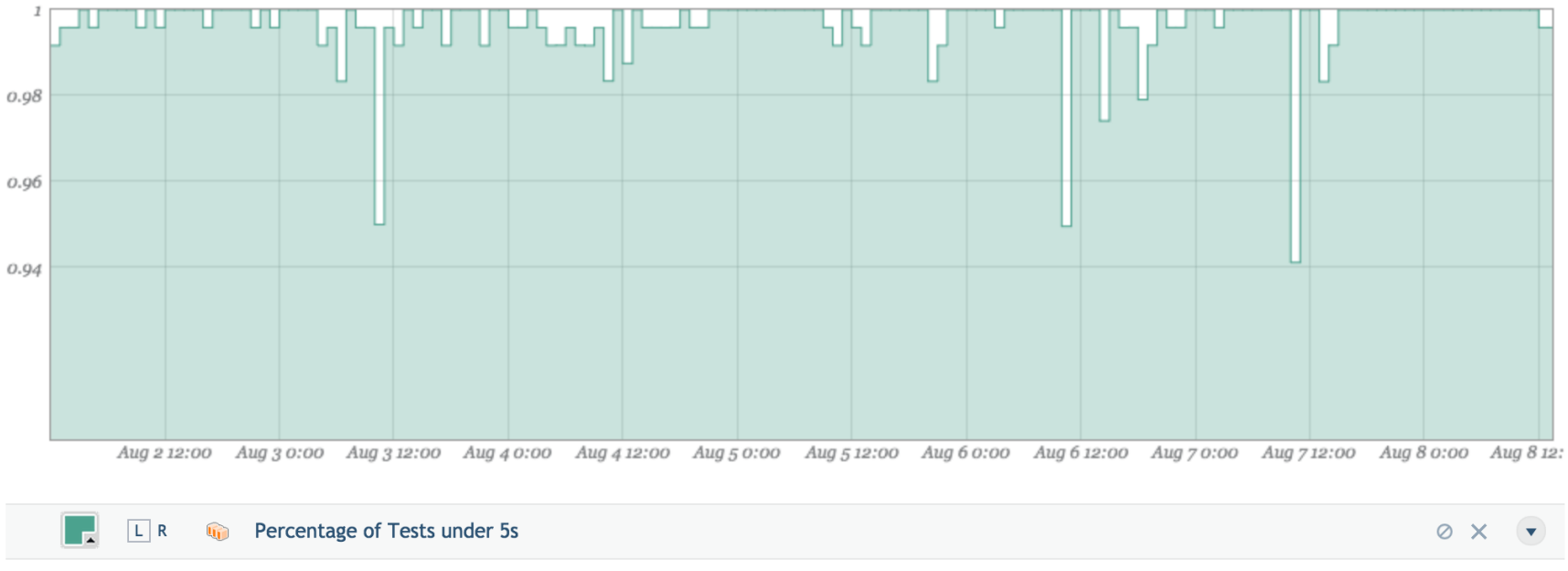
End-to-end testing



Averages are Poor for Anomaly Detection

- SLAs are typically not represented in terms of averages
- Averages combine both frequency and severity in one metric:
  - Outliers can magnify the appearance of error
  - Mild outliers can be lost
- It's important to understand both spread of your data and rate of occurrence of unacceptable events.

End-to-end testing (Inverse Quartile)



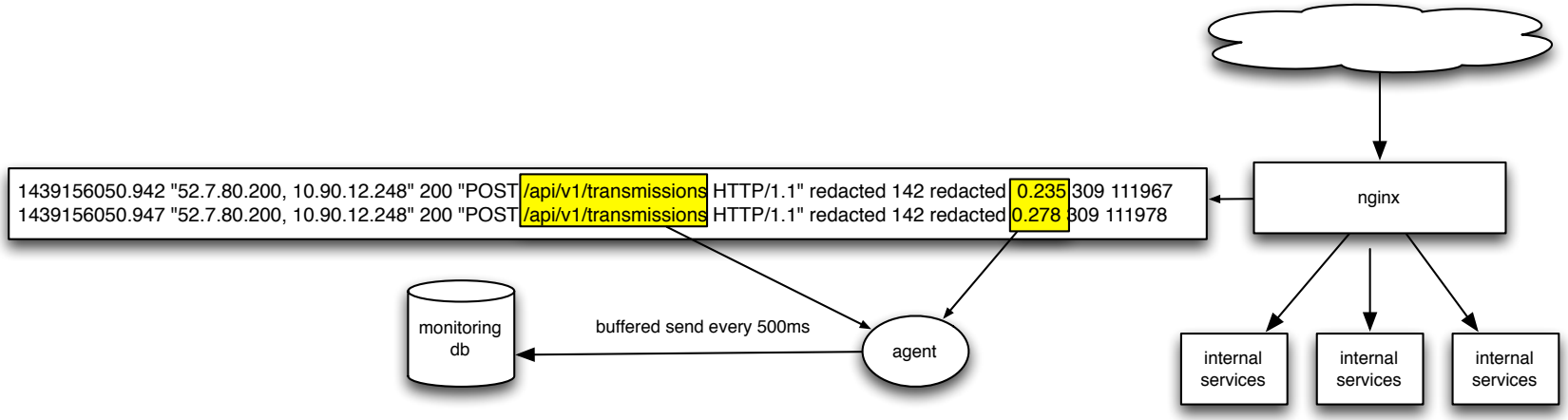
Passive Monitoring

- Capture data on every event
- Pass application information through an ETL into your monitoring datastore
- Some examples:
  - Real time indexing of logs
  - Real time extraction of log stream information and micro-batch inserts
  - Retooling of your application to collect real-time data and push directly into your monioring datastore

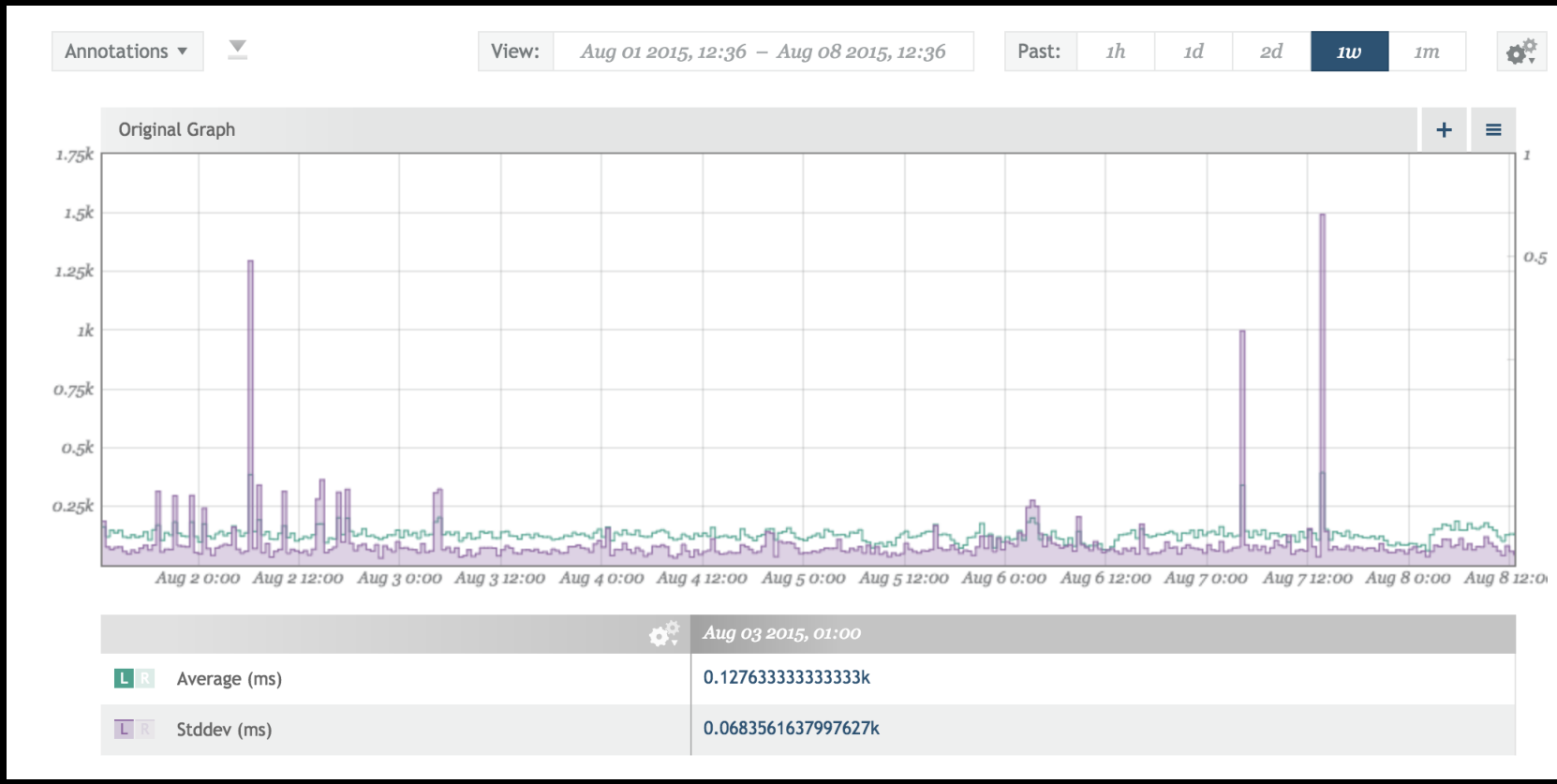
Passive Monitoring

- Full population data
- Pros
  - Collect data from actual usage
  - Data can come from existing application sources
    - Log files / custom application extensions
  - Possible to log very deep internal server side data
  - Gives you access to the data on every API interaction
- Cons
  - Less data depth compared to active monitors (can't measure client side)
  - Monitoring complex workflows is harder
  - May require work to be non-invasive

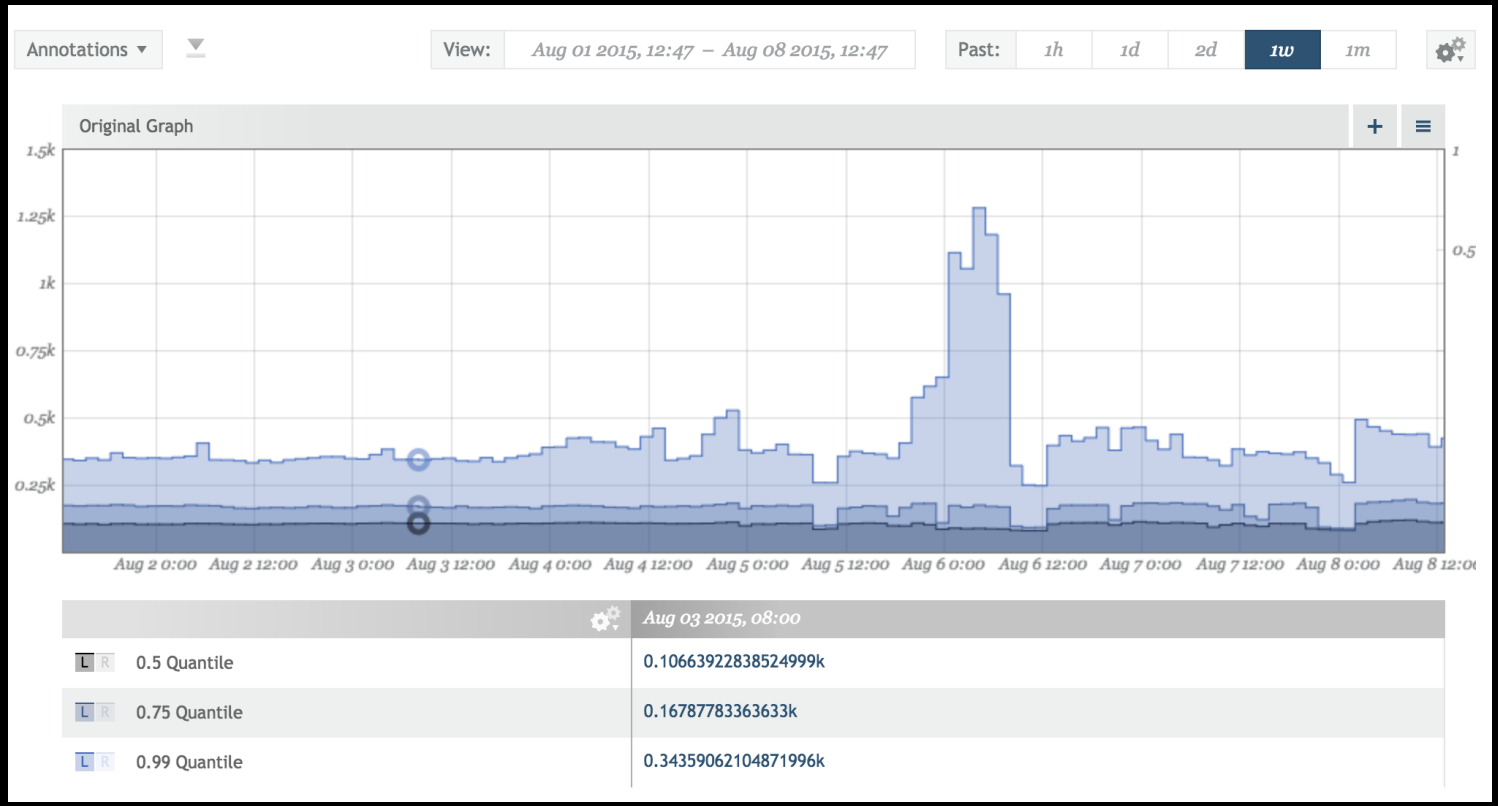
Our ETL for passive API Monitoring



Average API Latency (and stddev) – passive colelction

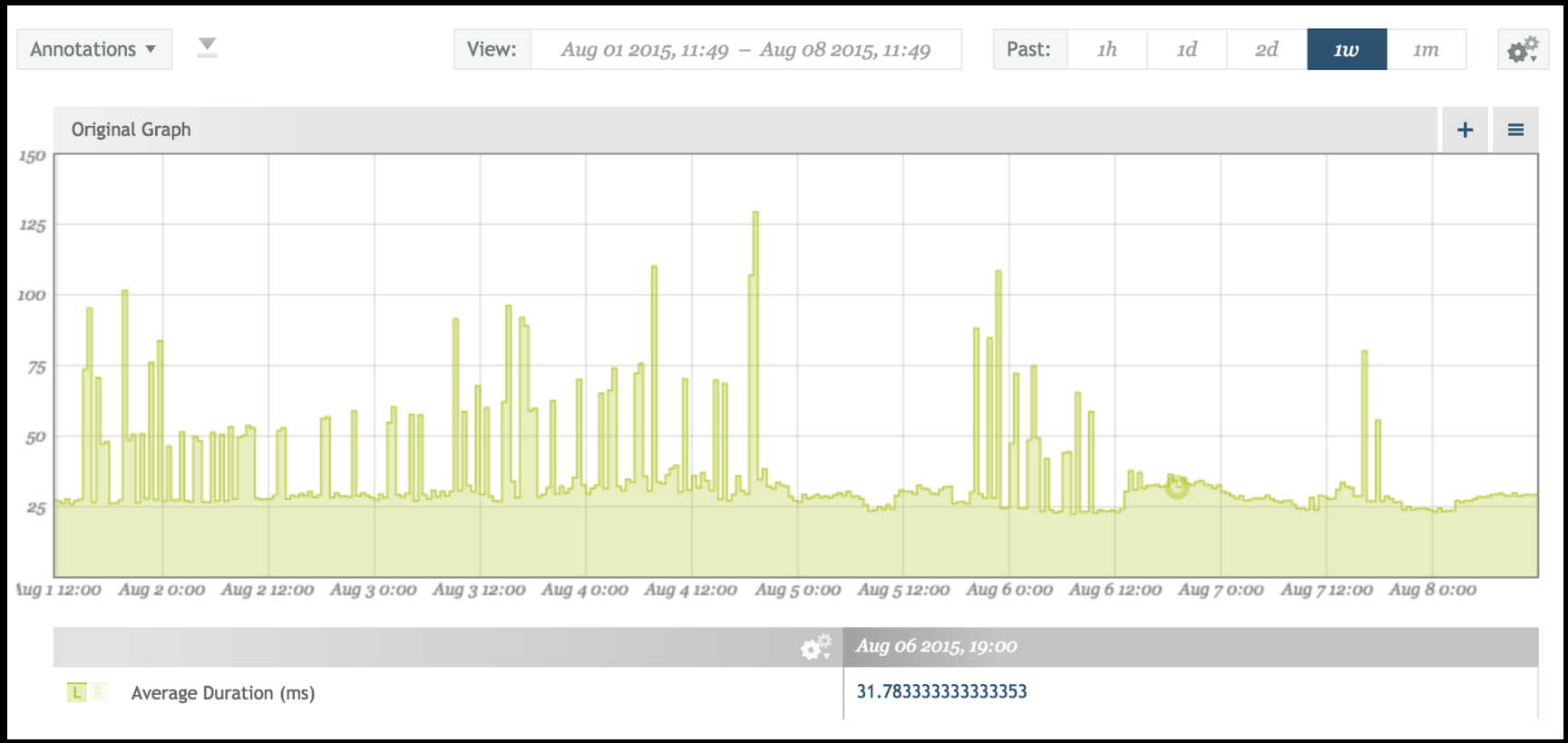


Quartile/Percentile Analysis

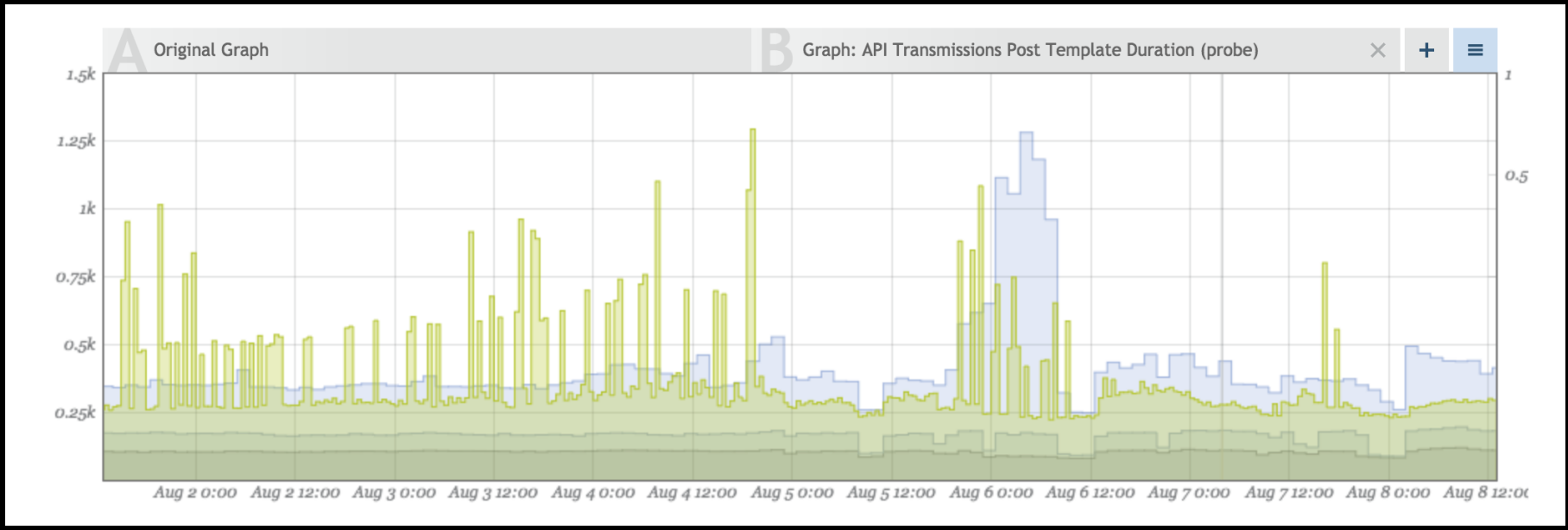


Inverse Quartile Analysis for SLA Compliance

Reminder of What Our Simple Check Looked Like



Comparing Active and Passive Results



Summary / Takeaways

- Active monitoring is very good at service availability monitoring, functional validation and general performance trending
- Active monitoring is not necessarily a good representation of customer experience
- Using a combined strategy provides the best picture of the health of your services

Some Additional Topics

- Local vs Off-Network Monitoring
  - Local removes network effects
  - Remote injects network effects
- Probablistic Sampling
  - Particularly useful where heavy-weight profiling is too intensive for full workload
- Passive error tracking of errors
  - Logfile analysis
  - Response code frequency analysis

Thanks! Questions?

Some links:

http://messagesystems.com

http://sparkpost.com