Continuous Performance Testing: The New Standard

Performance Paradigm Shift

4/9/2014

ObbiePet@livenation.com

1 FootnoteH

SOURCE: Source

Five years ago – the test request

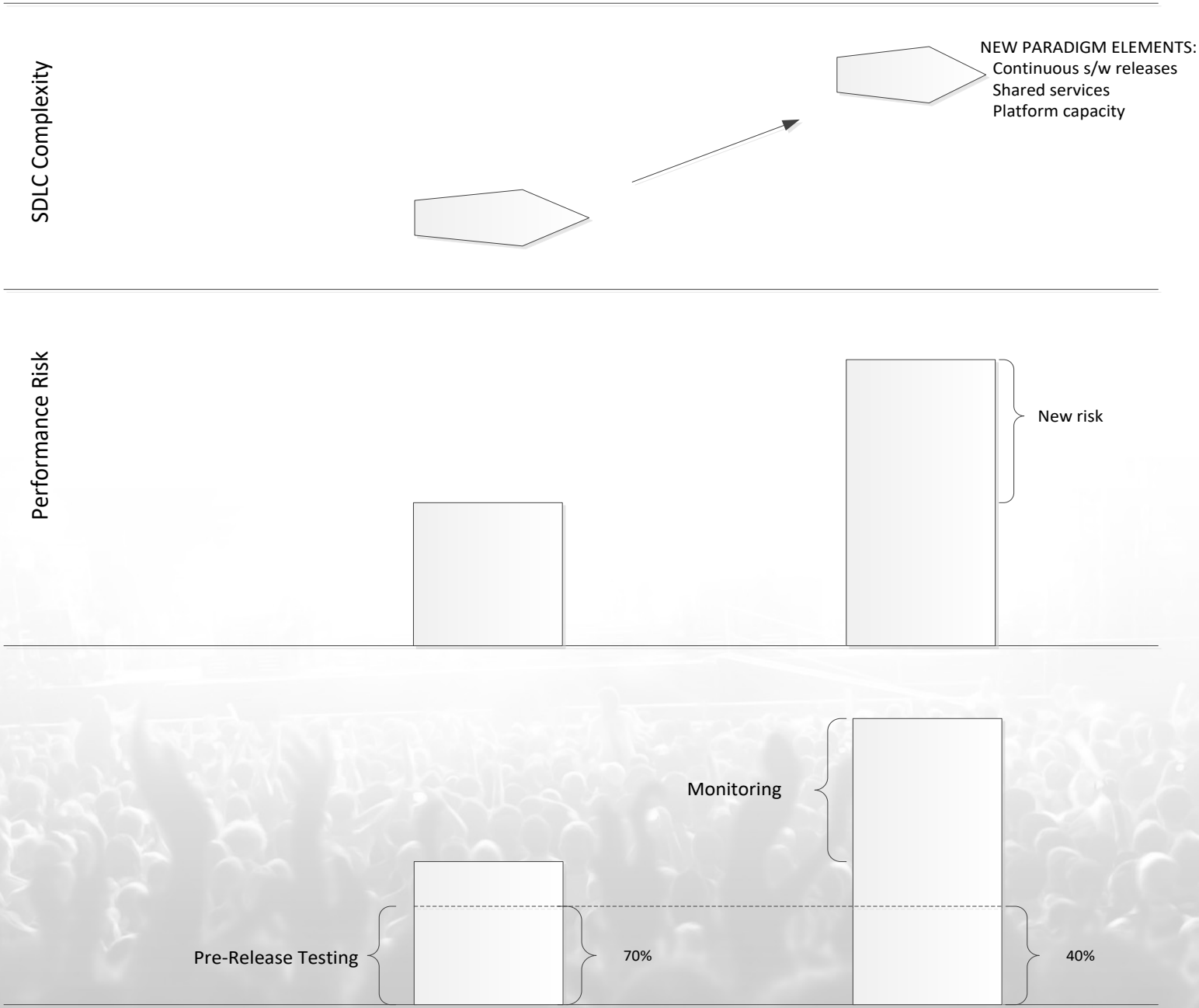
- 3rd party vendor test for ticket purchase app
- Register user hits via 3rd party servers (Dart)
- Performance test passed – Test servers work
- Does this give confidence in production.
- User experience is as good as the weakest link
- OnSale a month after the test,
  - Confident in our stuff, 3rd party stuff??
  - Tested code still the same?
  - Who else is using their site
- I am not confident!

1 FootnoteH
SOURCE: Source

Raising performance risk and its mitigation thru Monitoring

- A new paradigm in application development has unfolded over the last several years. Elements of this new paradigm introduce new performance risks. To assure good performance in production, these new risks need to be understood and mitigated.

1 FootnoteH
SOURCE: Source



1 FootnoteH

SOURCE: Source

Important takeaway

If you walk away with only one thing from this meeting…

… performance monitoring has become essential to assuring production performance.

A 2nd arrow has been added to our quiver.

Arrow 1: Prelease testing

Arrow 2: Production monitoring ← NEW

Over the remaining slides, I will:

1) Layout the case for monitoring
2) Offer a practical approach to implement performance assurance which includes monitoring.

1 FootnoteH

SOURCE: Source

Theory

1 FootnoteH
SOURCE: Source

Why do we performance test?

- To improve and maintain performance in PRODUCTION

1 FootnoteH
SOURCE: Source

How do we assure good performance in Production?

- Old Paradigm / Traditional Approach
  - Software released quarterly or monthly.
  - Changes between releases are rare.

- **How do we assure good performance in this traditional environment?**

1 FootnoteH
SOURCE: Source

How do we assure good performance in the traditional Production environment?

- Pre-Release testing
  - Production candidate code is deployed to a staging environment, similar to production.
  - High demand is simulated in performance tests and problems are detected and fixed before production deployment.
  - Once fixed, the system is stable until the next release.
- ASSUMPTION: The production environment is stable between releases.

1 FootnoteH
SOURCE: Source

How do we assure good performance in the traditional Production environment?

- Pre-Release testing
- ASSUMPTION: The production environment is stable between releases.
- Is this assumption still valid?

1 FootnoteH
SOURCE: Source

Is production stable between releases?

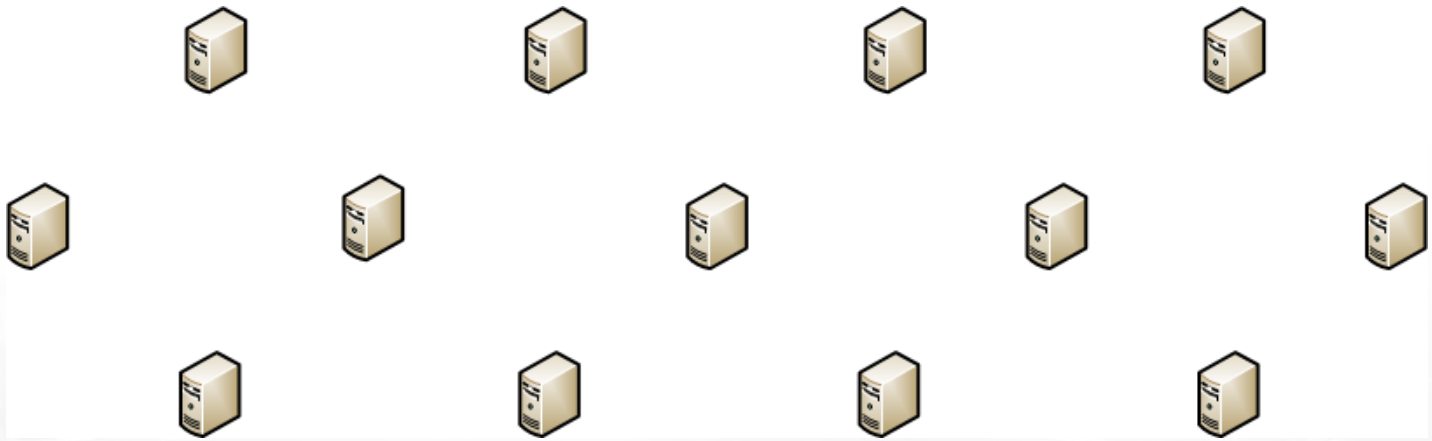
- Let's look at modern SDLC application characteristics and see.

Application characteristics	Old Paradigm	New Paradigm
s/w release frequency, frequency of s/w component changes	+----+----+----+----	++++++++++++
Shared services	Minimal, predictable demand	Extensive, unpredicatable demand
Platform capacity	Fixed, dedicated h/w	Variable, cloud implementation

1 FootnoteH

SOURCE: Source

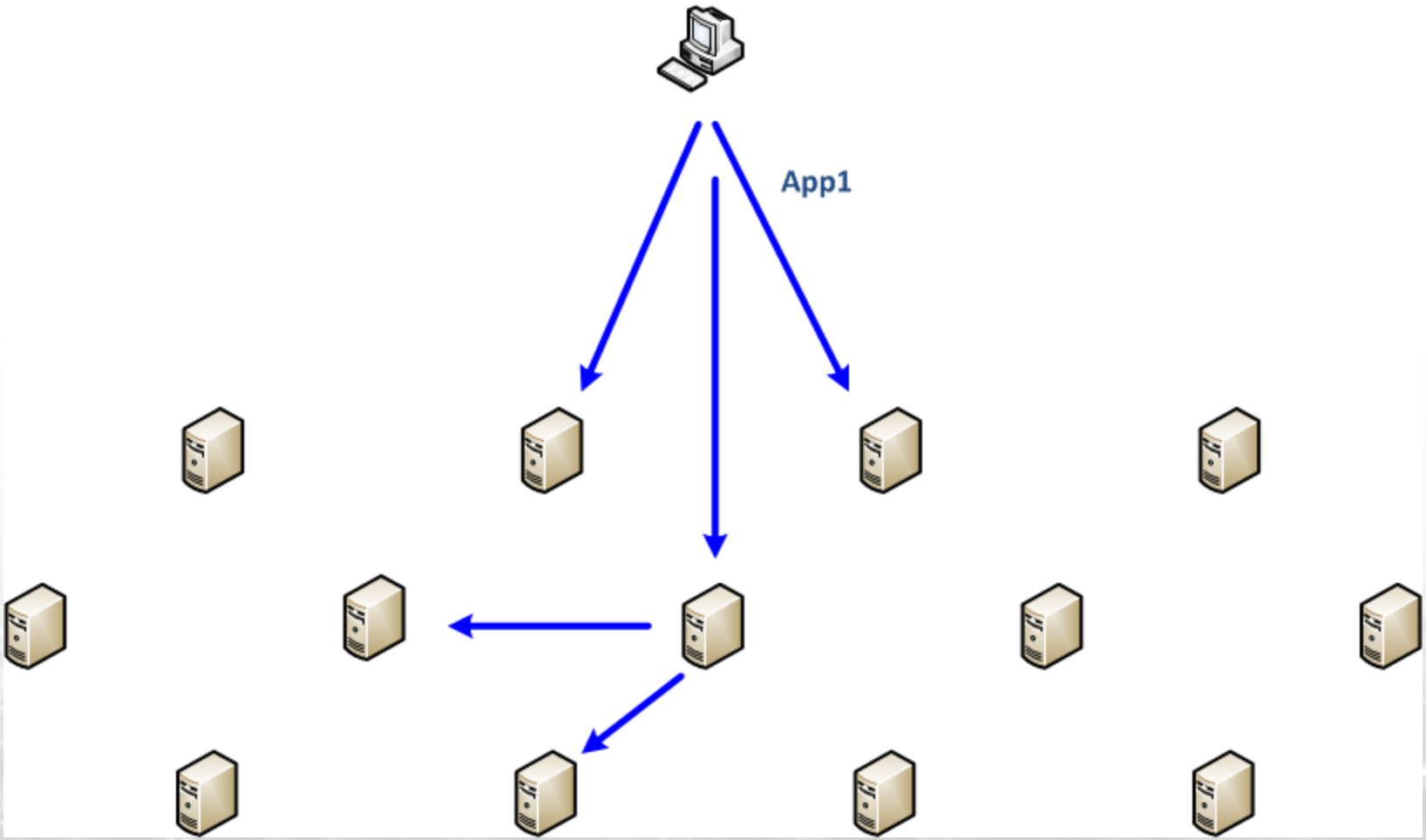
Shared services – explained 1 of 5



1 FootnoteH

SOURCE: Source

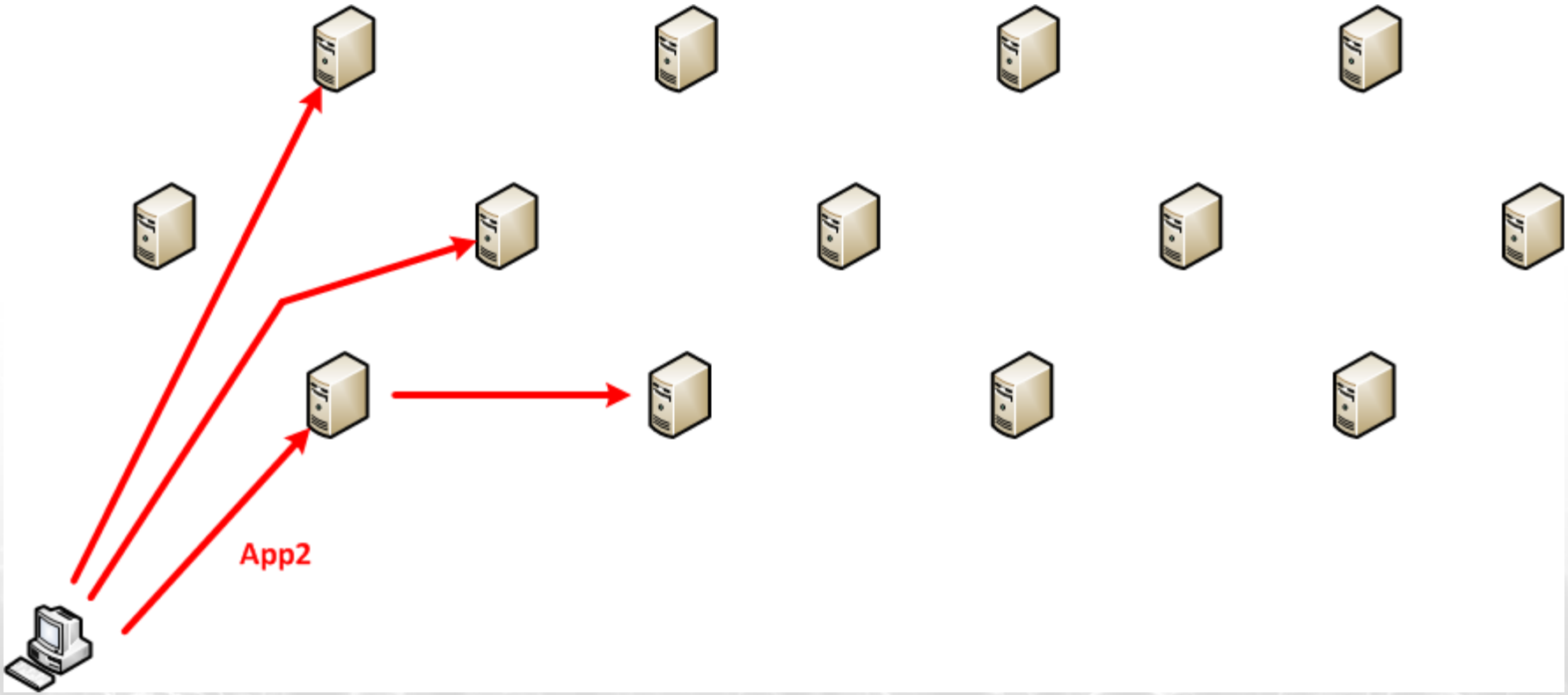
Shared services – explained 2 of 5



1 FootnoteH

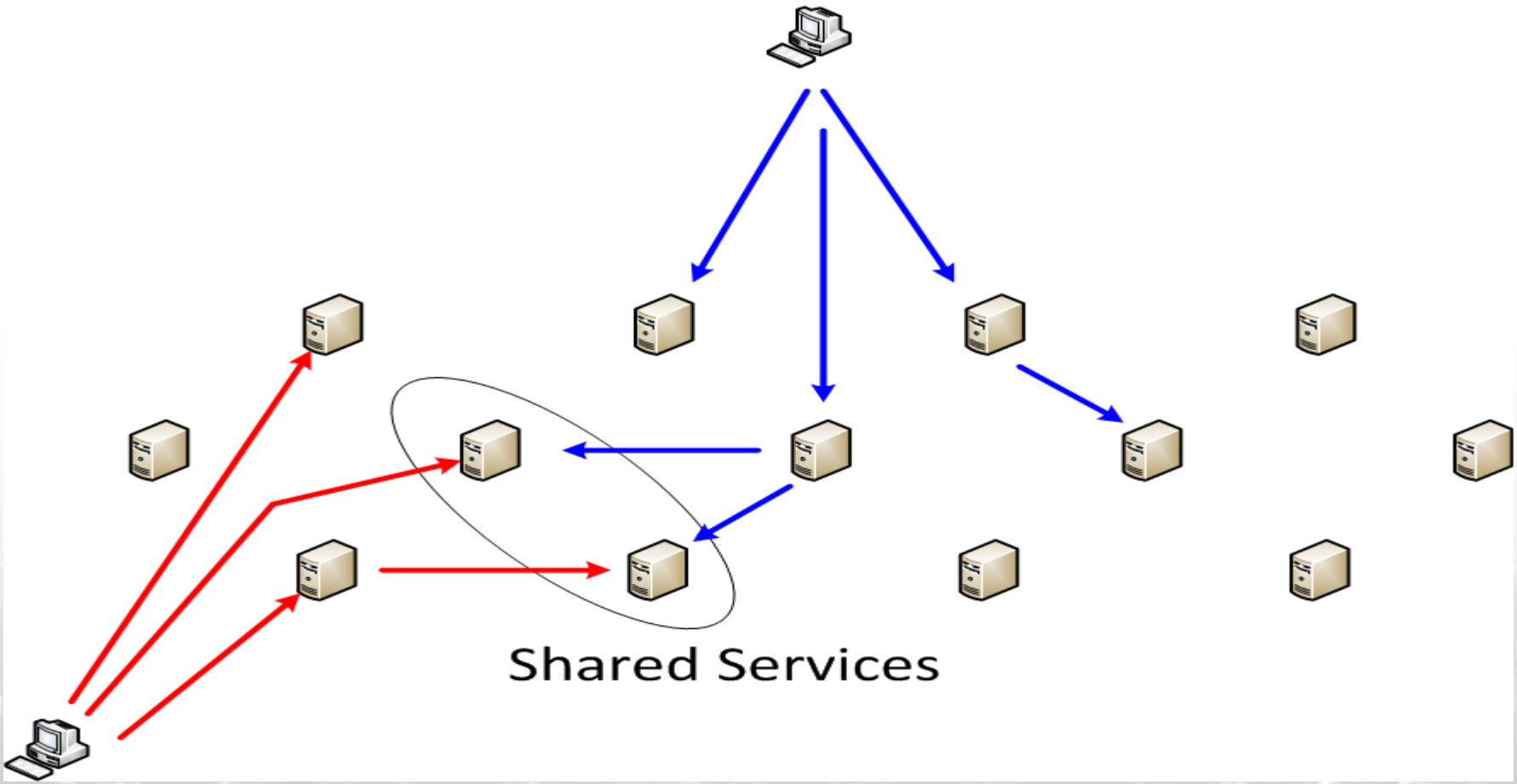
SOURCE: Source

Shared services – explained 3 of 5



1 FootnoteH
SOURCE: Source

Shared services – explained 4 of 5



1 FootnoteH
SOURCE: Source

Shared services – explained 5 of 5

- Imagine 5, 10, or 50 applications concurrently running.
- It becomes hard to predict the demand on shared services.

1 FootnoteH
SOURCE: Source

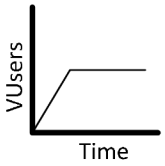
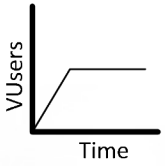
Platform capacity– explained 1 of 6

Test Scenario + Code Base + H/W = Test Results

1 FootnoteH

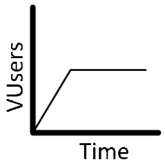
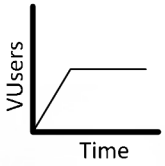
SOURCE: Source

Platform capacity– explained 2 of 6

Test Scenario	+	Code Base	+	H/W	=	Test Results
	+	Before Change	+	 PerfB 4 CPU	=	Baseline
	+	Afer Change	+	 PerfB 4 CPU	=	Benchmark

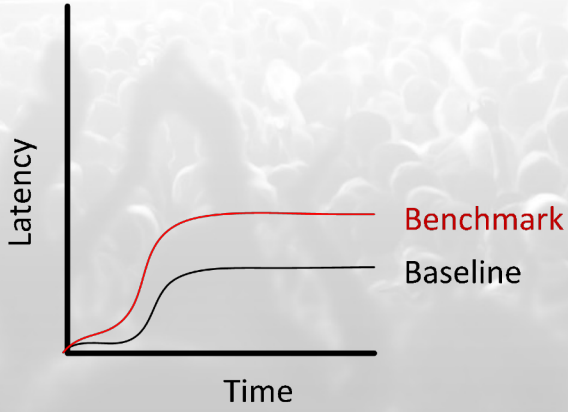
1 FootnoteH
SOURCE: Source

Platform capacity– explained 3 of 6

Test Scenario	+	Code Base	+	H/W	=	Test Results
	+	Before Change	+	 PerfB 4 CPU	=	Baseline
	+	Afer Change	+	 PerfB 4 CPU	=	Benchmark

Did the Benchmark's code change make performance worse?

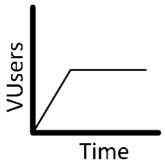
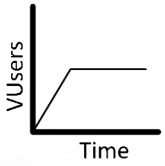
Note: Only one variable changed between tests. We can correlate any performance change to that variable.



1 FootnoteH

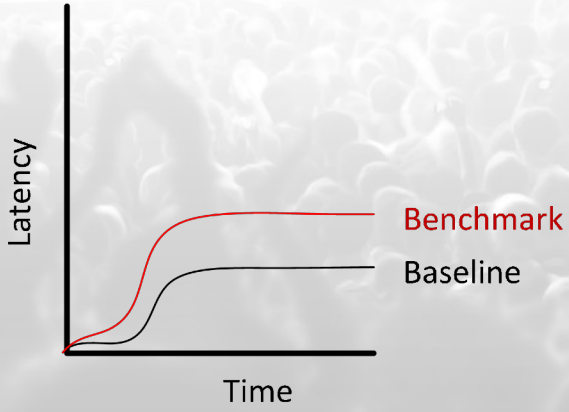
SOURCE: Source

Platform capacity– explained 4 of 6

Test Scenario	+	Code Base	+	H/W	=	Test Results
	+	Before Change	+	 PerfB 4 CPU	=	Baseline
	+	Afer Change	+	 Dev 2 CPU	=	Benchmark

Did the Benchmark's code change make performance worse?

How many variables changed between tests?
Which was responsible for the performance delta?

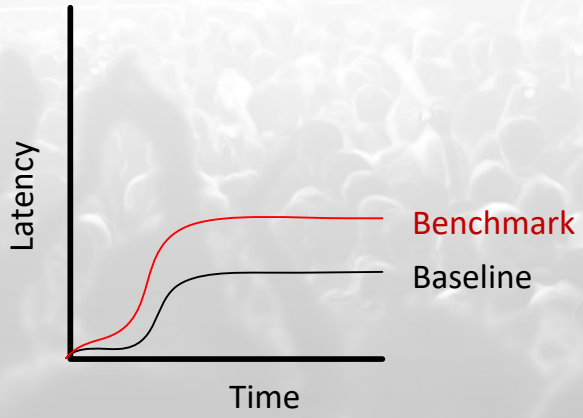


1 Footnote1
SOURCE: Source

Platform capacity– explained 5 of 6

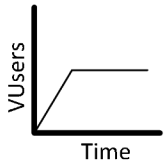
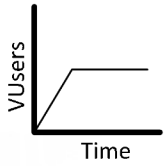
The VM/Cloud problem.

Did the Benchmark's code change make performance worse?



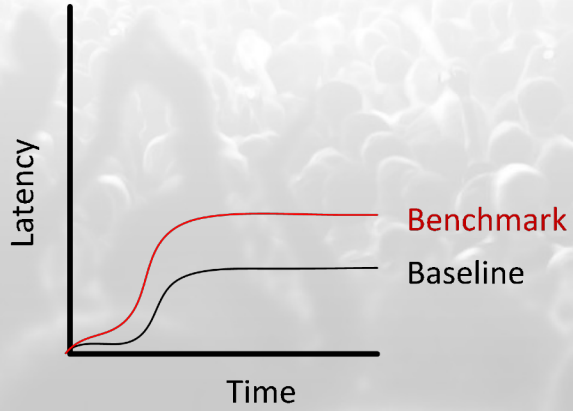
1 FootnoteH
SOURCE: Source

Platform capacity– explained 6 of 6

Test Scenario	+	Code Base	+	H/W	=	Test Results
	+	Before Change	+	 VM ?? CPU	=	Baseline
	+	Afer Change	+	 VM ?? CPU	=	Benchmark

Did the Benchmark's code change make performance worse?

- 2 variables change between tests;
- intra test variability (the VM problem)



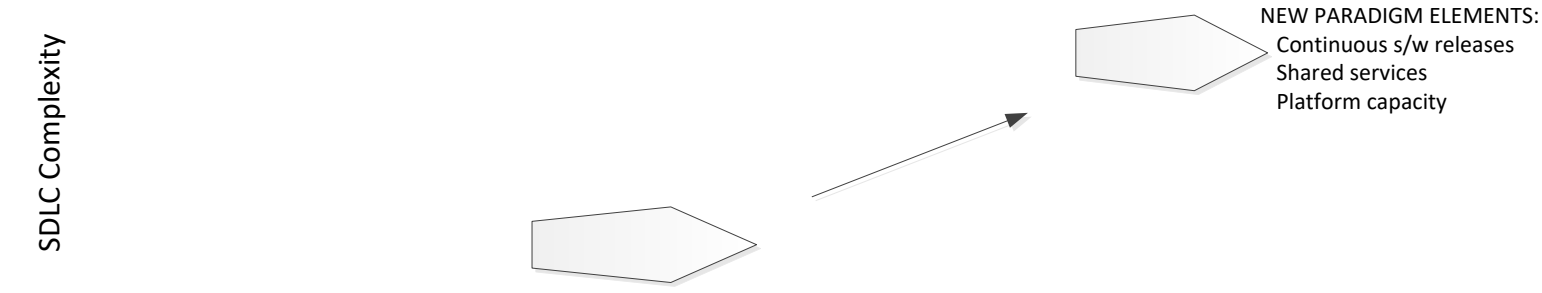
1 Footnote!!
SOURCE: Source

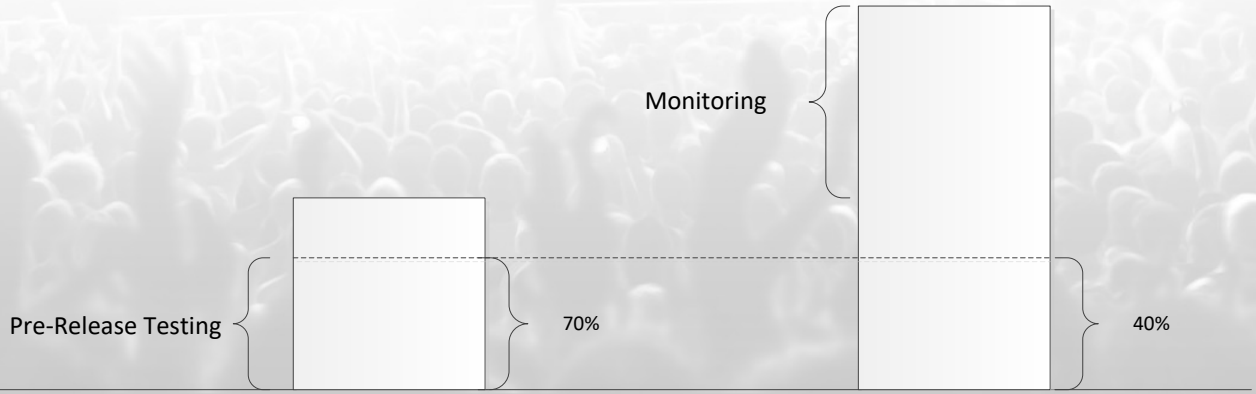
Is production stable between releases?

Application characteristics	Old Paradigm	New Paradigm
s/w release frequency, frequency of s/w component changes	+----+----+----+----	++++++++++++
Shared services	Minimal, predictable demand	Extensive, unpredictable demand
Platform capacity	Fixed, dedicated h/w	Variable, cloud implementation

- No.

1 FootnoteH
SOURCE: Source





1 FootnoteH

SOURCE: Source

How do we assure performance in such a dynamic environment?

- **AUGMENT PRE-PRELEASE TESTING WITH MONITORING.**
  - Monitoring, most importantly performance monitoring, can mitigate performance risk associated with the new paradigm.

1 FootnoteH
SOURCE: Source

How does monitoring mitigate these new risks?

Application characteristics	New Paradigm	Risk Mitigation via Monitoring
s/w release frequency, frequency of s/w component changes	++++++++++++	Provide immediate feedback when a s/w change has impacted a component.
Shared services	Extensive, unpredictable demand	Demand patterns on all services are observed, and can be managed.
Platform capacity	Variable, cloud implementation	Metrics used to prevent application starvation of CPU/Memory/Disk/ Network

1 FootnoteH

SOURCE: Source

What are the limitations of monitoring?

- Monitoring doesn't remove performance risks like Pre-Release testing. But it does put us in a position to minimize performance issues in production.
- Reactive vs proactive.
- It's the only technique I can think of to address these new risks.

- Lets see how performance monitoring works…

1 FootnoteH
SOURCE: Source

How does (Production) monitoring reduce performance risk?

- Prevent performance issues from entering production (when monitoring tools are used with Pre-Release testing)
- Early detection/remediation of performance issues, before the customer notices.
- Fast resolution of performance issues reported by customers.

1 FootnoteH
SOURCE: Source

How does (Production) monitoring reduce performance risk?

- *Example Performance problem:*
- Customers are calling in complaining the web site is taking too long to respond.
- Where is the problem? How does IT respond?

Assemble a team of experts from each tier (bad)

VS

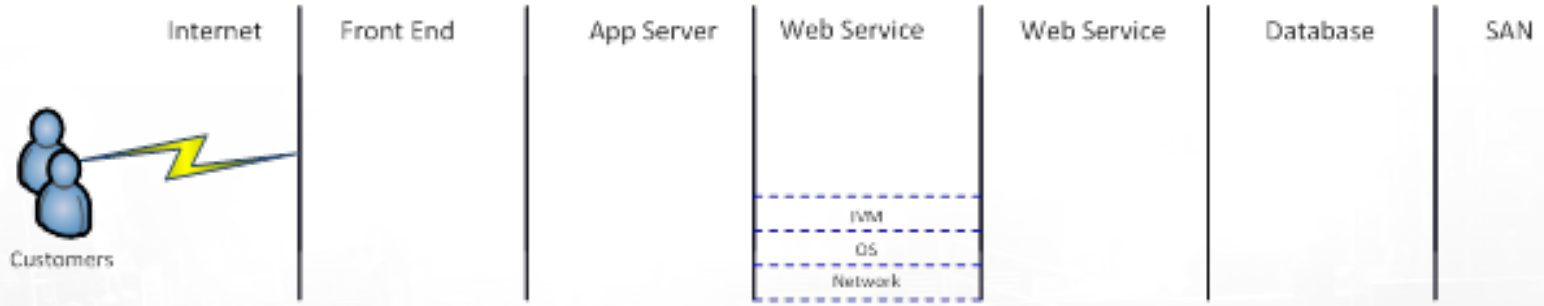
Monitored metrics immediately identify the broken tier (good).

1 FootnoteH

SOURCE: Source

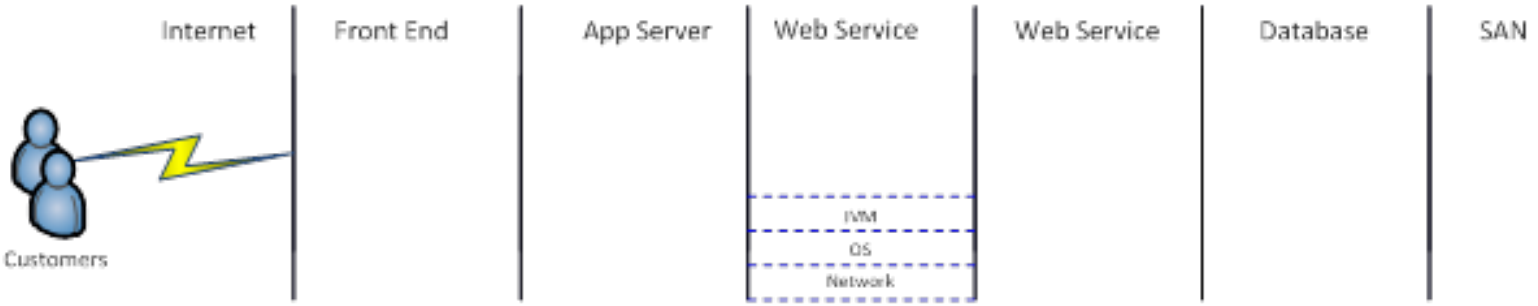
How does (Production) monitoring reduce performance risk?

- *Diagnostic strategy:*
- First isolate problem tier: left to right ↔ ;
- than isolate problem layer within the tier: up and down ↕



1 FootnoteH
SOURCE: Source

How does (Production) monitoring reduce performance risk?



- What metrics do we measure?
  - Performance (business) metrics
    most important, hardest to capture
  - Resource metrics
  - VM metrics

1 FootnoteH
SOURCE: Source

What metrics do we measure?

- What metrics do we measure?
  - Performance (business) metrics for tier isolation most important, hardest to capture
    - Transaction response times
    - Transaction rates
    - Error rates
  - Resource metrics for layer isolation
    - CPU
    - Memory
    - etc…
  - VM metrics

1 FootnoteH
SOURCE: Source

Monitoring Examples –
What do monitoring dashboards look like?

- Examples provided use open source tools and methodology.
- Kibana
- Open TSDB

- Focus will be on performance monitoring (rates, resp times, and errors)

1 FootnoteH
SOURCE: Source

Monitoring Examples –
What do monitoring dashboards look like?

Target of test –
I'm going to talk about an email notification system, multiple services orchestrated into a product.

Performance dashboard examples from a test run.

Performance dashboard examples from production.

1 FootnoteH
SOURCE: Source

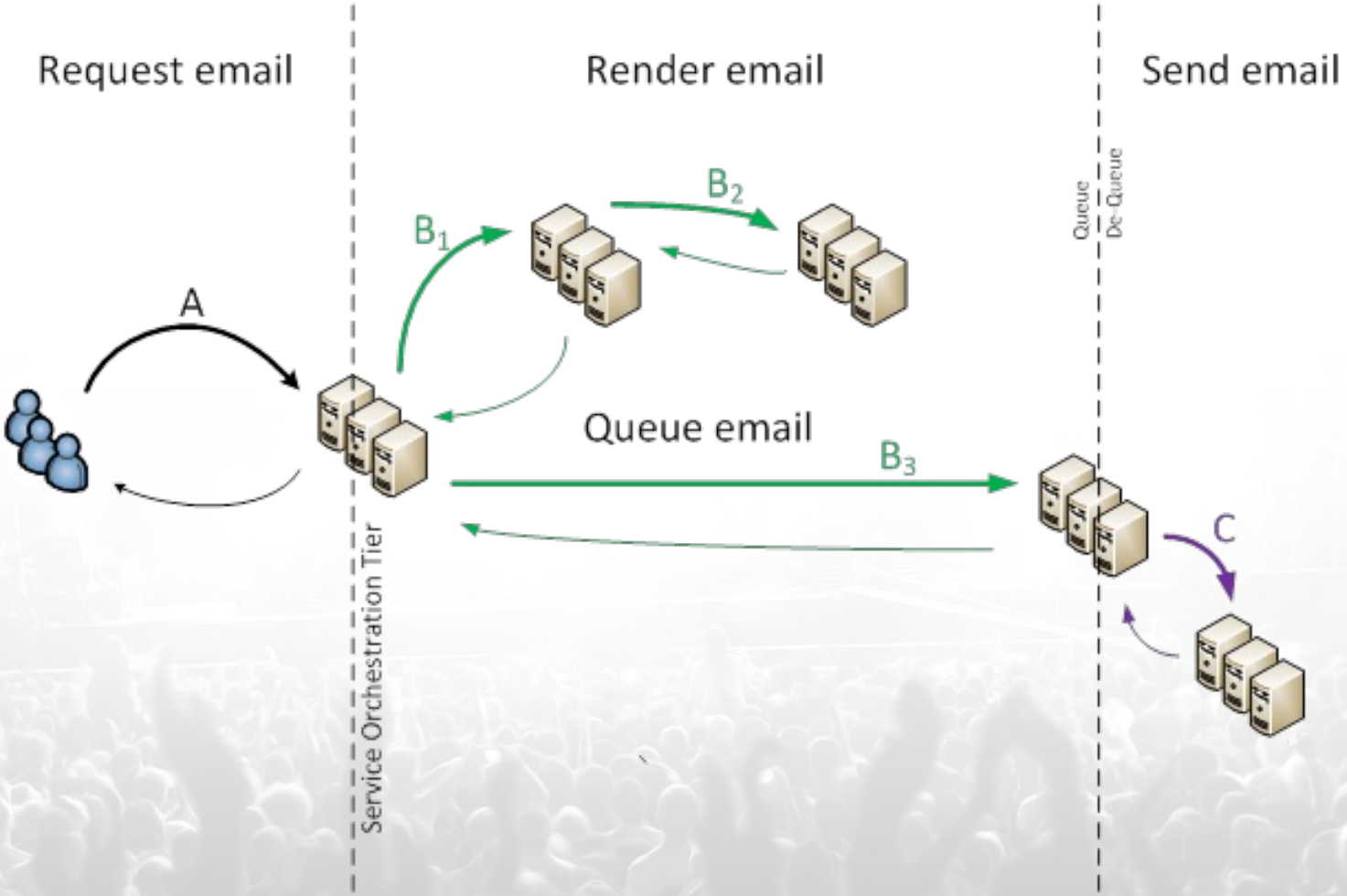
Monitoring Examples –
What do monitoring dashboards look like?

Target of test – email notification system
multiple services orchestrated into a product

- We want visibility into each service tier

1 FootnoteH
SOURCE: Source

Monitoring Examples – SOA architecture of an email notification system



1 FootnoteH
SOURCE: Source

Monitoring Examples –
Performance dashboard examples from a test run

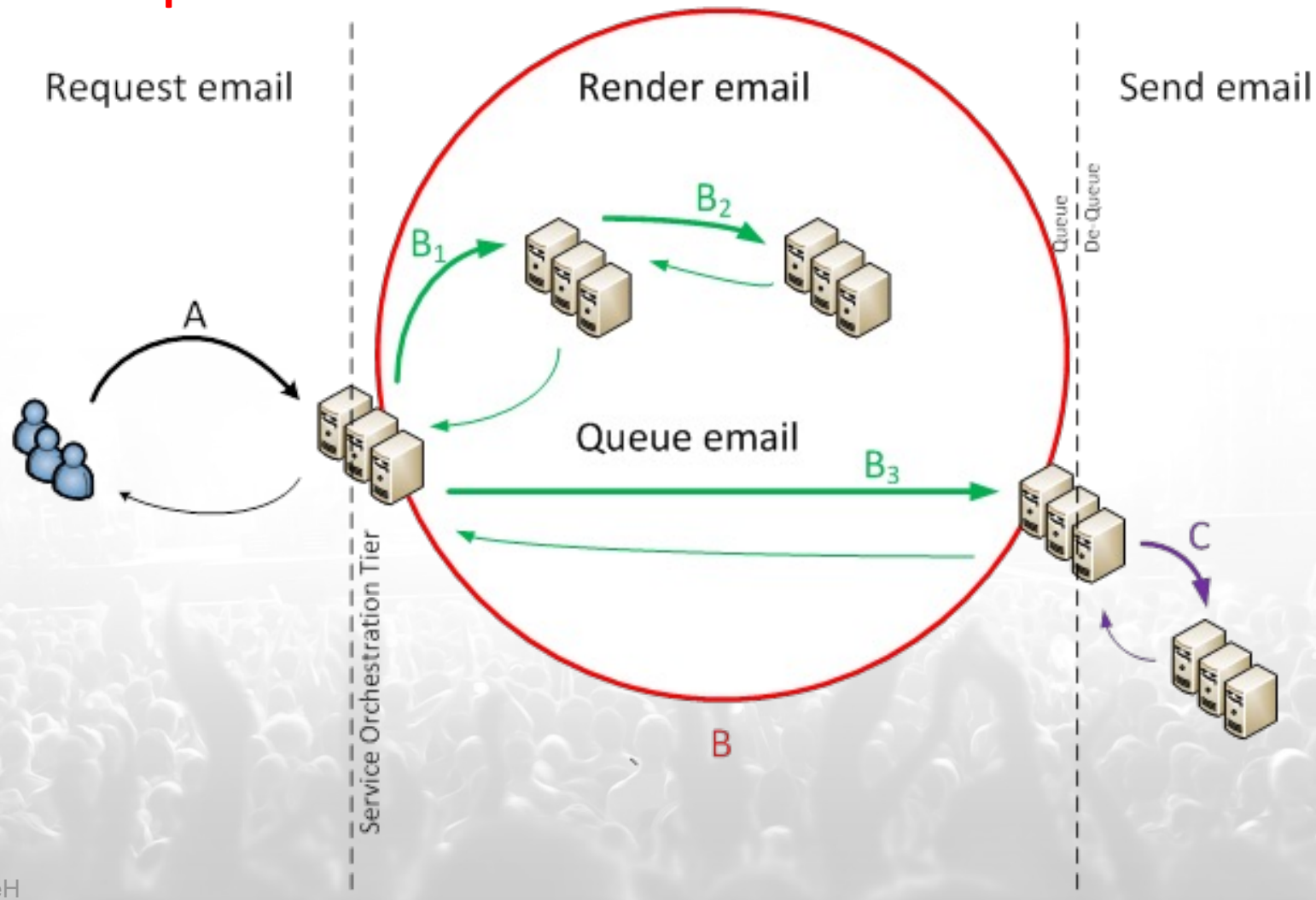
- Service dashboard examples from a healthy test
- A performance problem viewed from a dashboard
- Performance problem isolation via a dashboard

- Resource metric monitor example

1 FootnoteH
SOURCE: Source

Monitoring Examples –
Service dashboard examples from a healthy test

- **Composite transaction B**



1 FootnoteH
SOURCE: Source

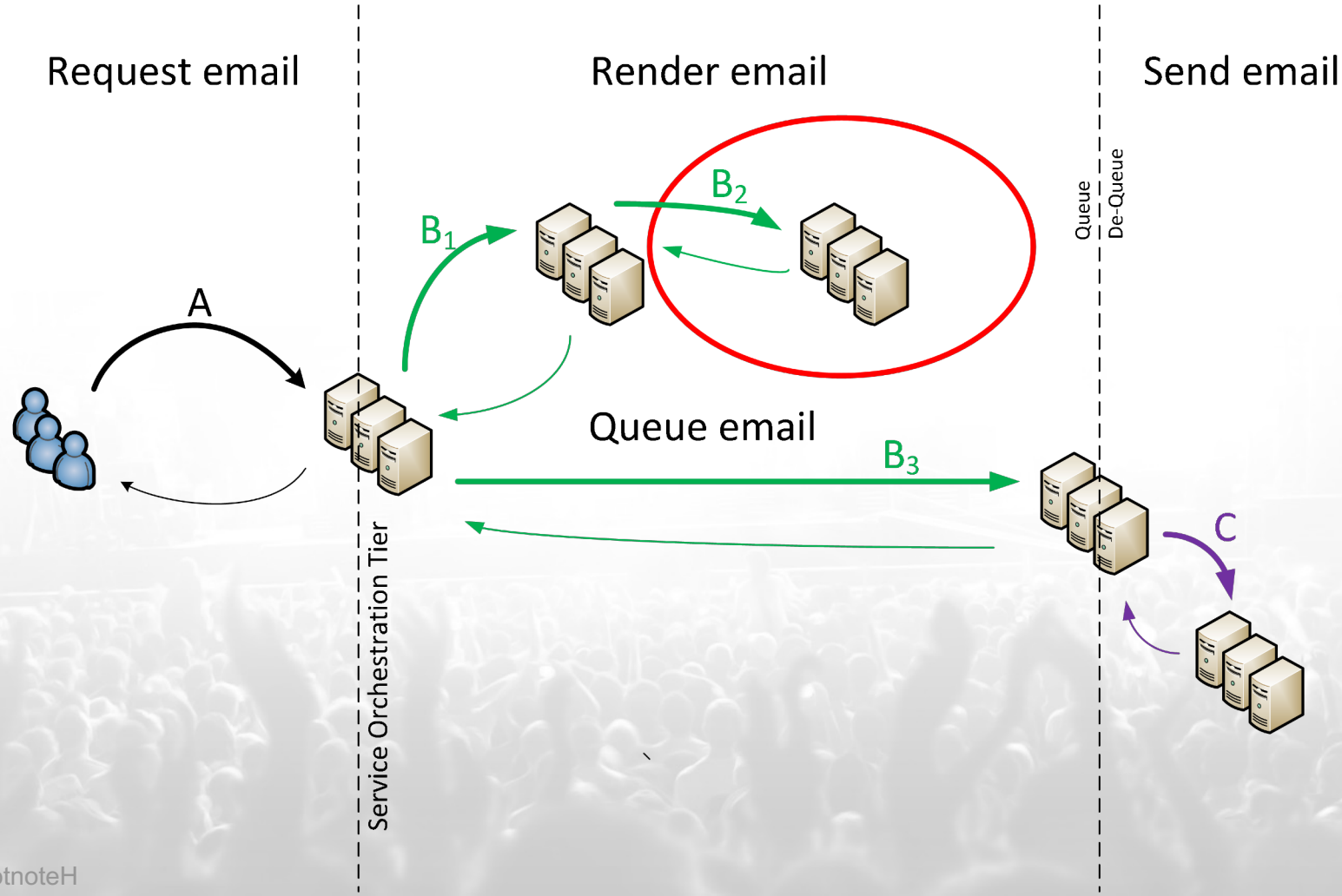
Monitoring Examples –
Service dashboard examples from a healthy test

- **Composite transaction B**

1 Foc
SOURCE: Source

Monitoring Examples –
Service dashboard examples from a healthy test

- **Composite transaction B_2**



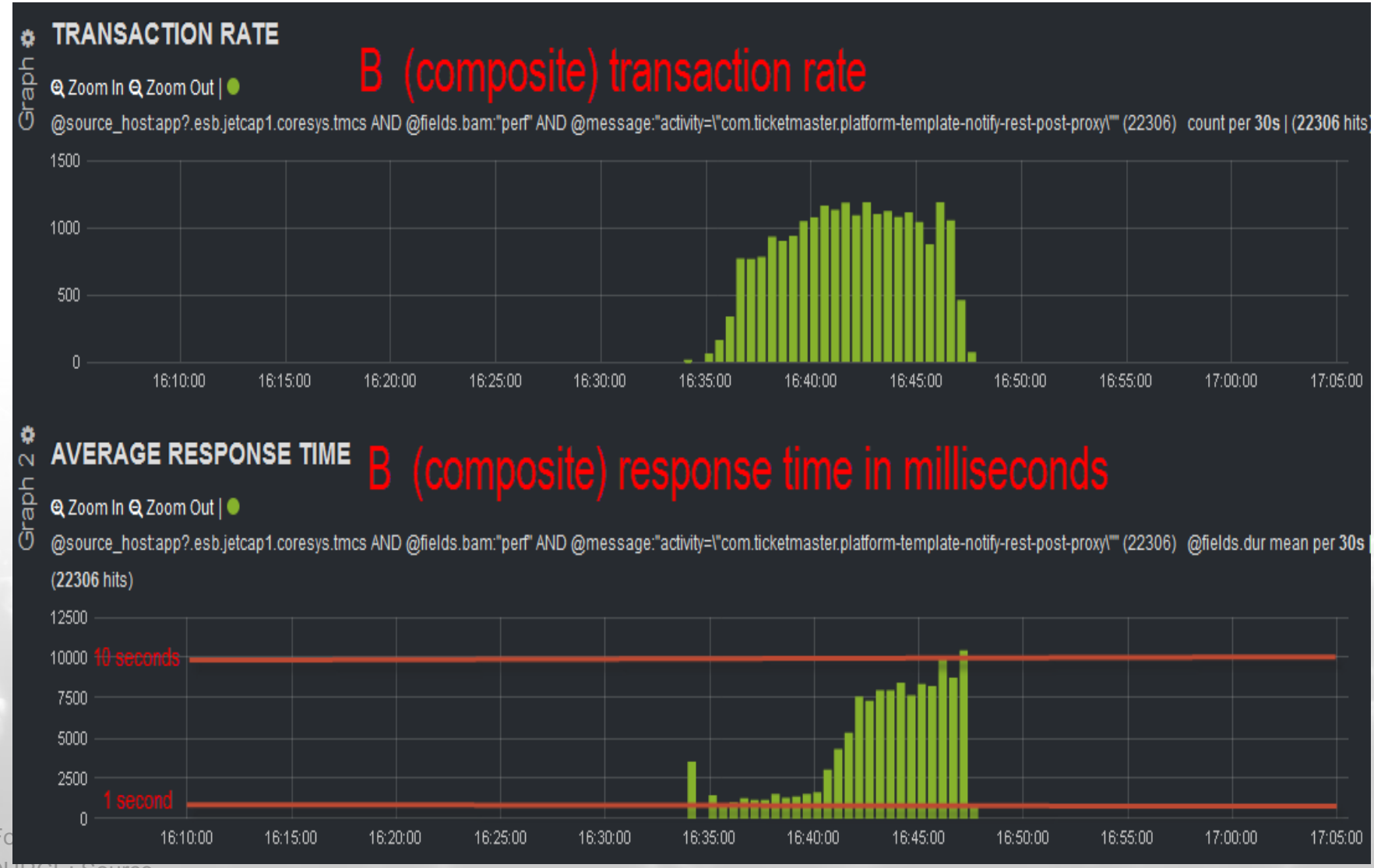
1 FootnoteH
SOURCE: Source

Monitoring Examples –
Service dashboard examples from a healthy test

- **Composite transaction B_2**

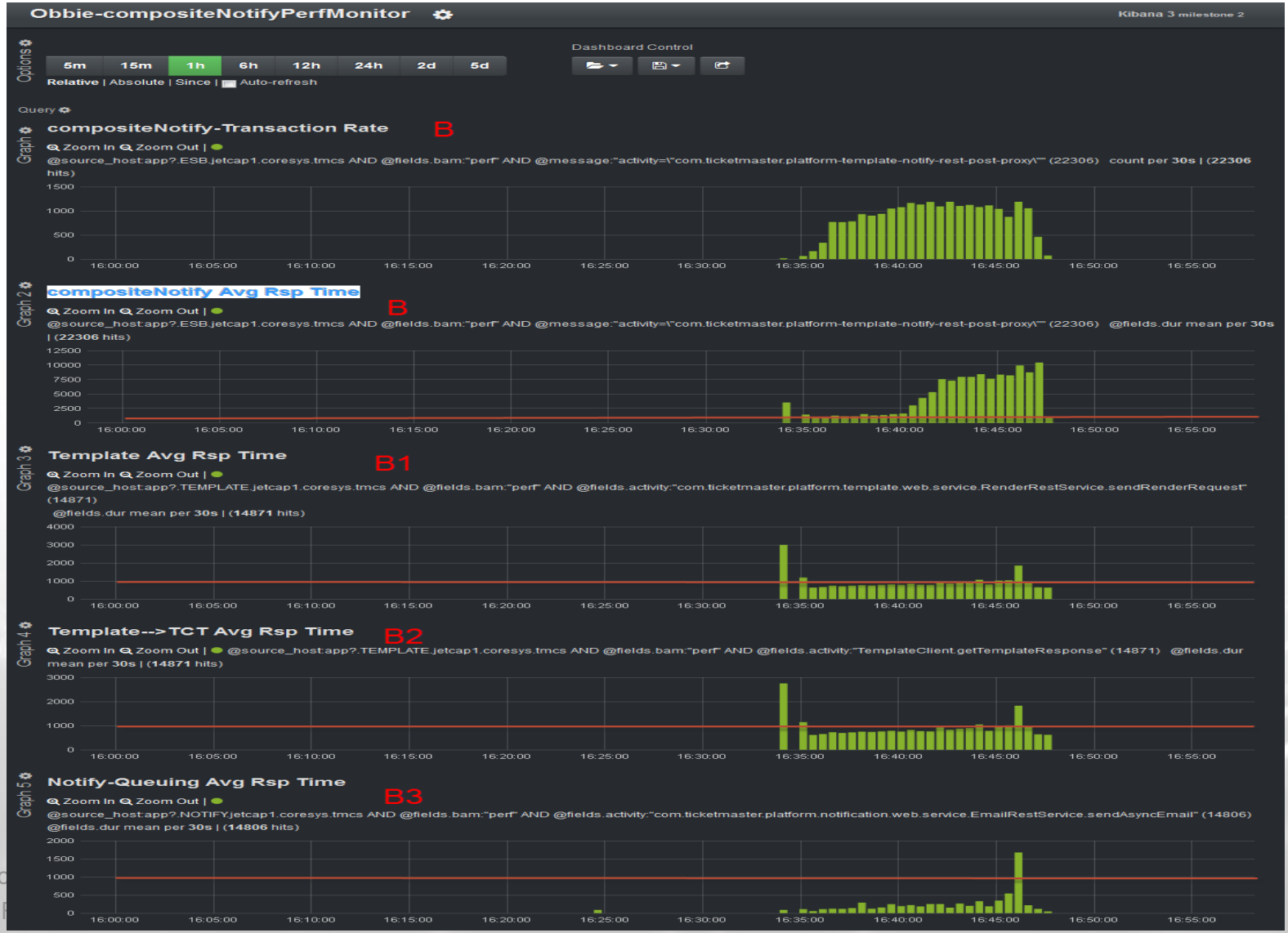
1 Foc
SOURCE: Source

Monitoring Examples –
A performance problem viewed from a dashboard

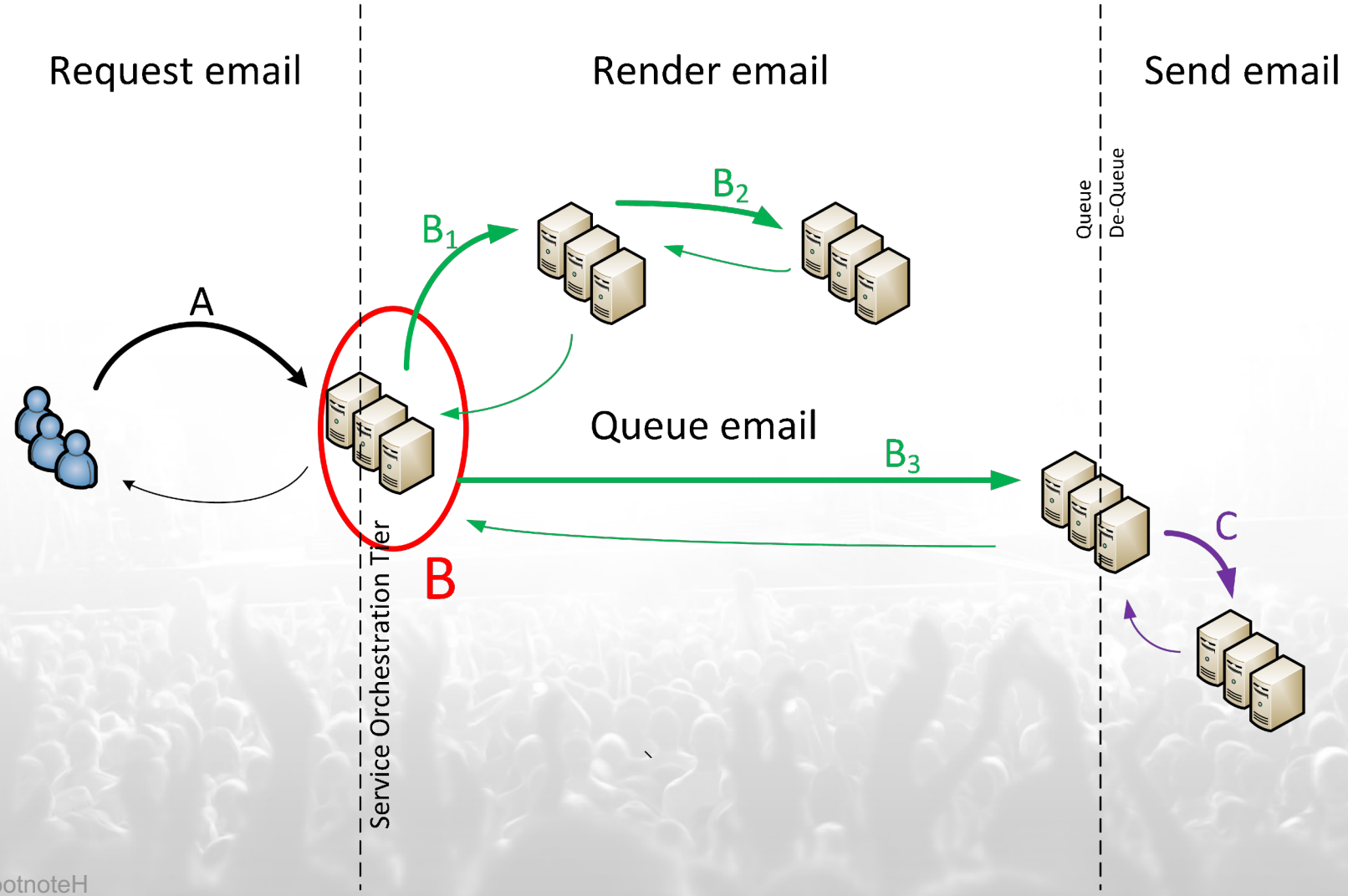


1 Fo
SOURCE: Source

Monitoring Examples –
Isolating the performance problem via a dashboard



Monitoring Examples – Isolating the performance problem via a dashboard



1 FootnoteH

SOURCE: Source

Monitoring Examples –
Resource metric monitor (Open TSDB)

Monitoring Examples – Performance dashboard examples from Production

- Two dashboards will be shown
  - Dashboard 1:

    How is the PDF rendering service performing?

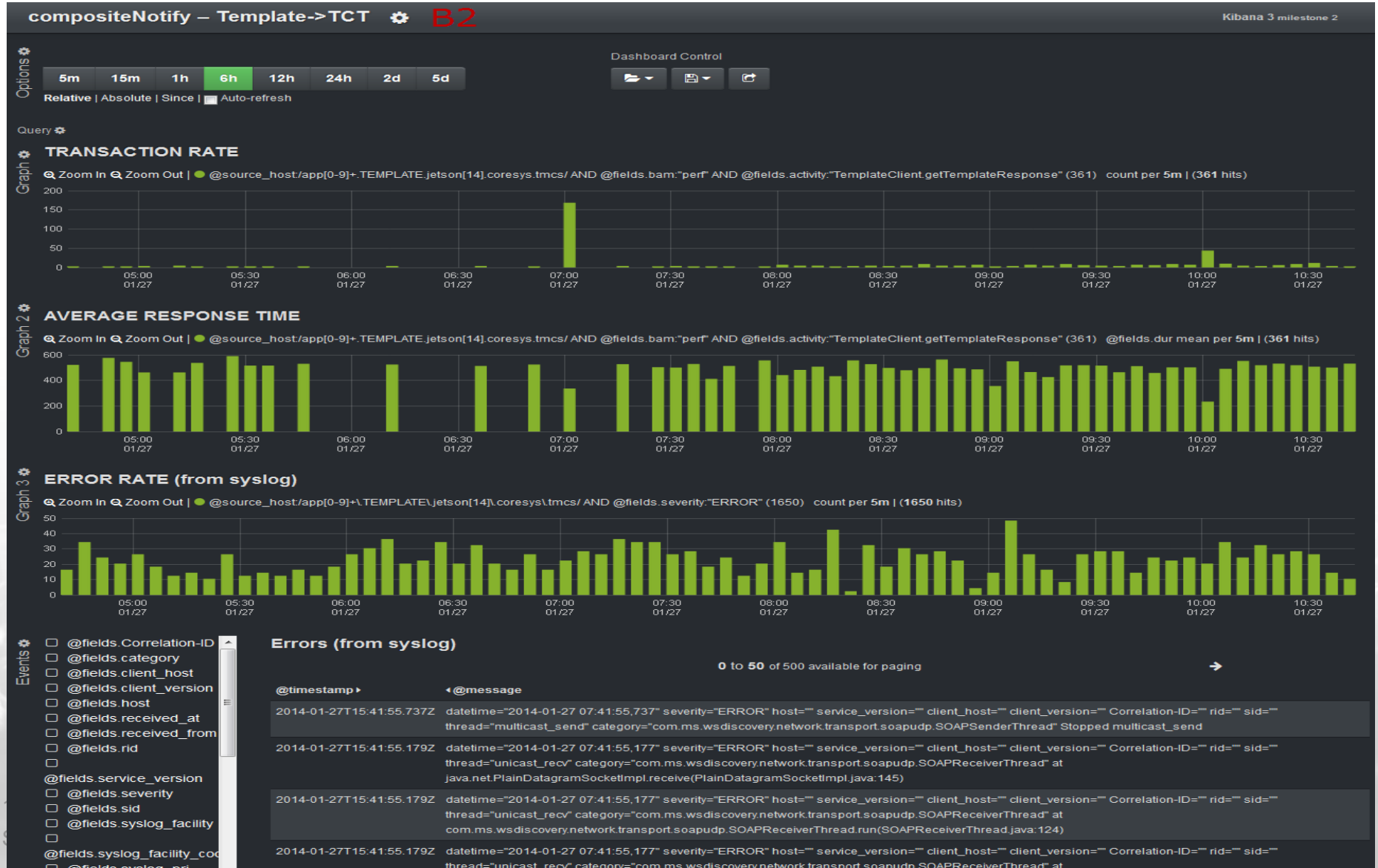
  - Dashboard 2:

    Is Production email being delivered within SLA's?

1 FootnoteH

SOURCE: Source

Monitoring Examples –
How is the PDF rendering service performing?



Monitoring Examples –
Is Production email being delivered within SLA's?

- See next slide

1 FootnoteH
SOURCE: Source

LIVE NATION ENTERTAINMENT

compositeNotify – PerformanceSummary

Kibana 3 milestone 2

Options

5m | 15m | 1h | 6h | 12h | 24h | 2d | 5d

Relative | Absolute | Since | Auto-refresh

Dashboard Control

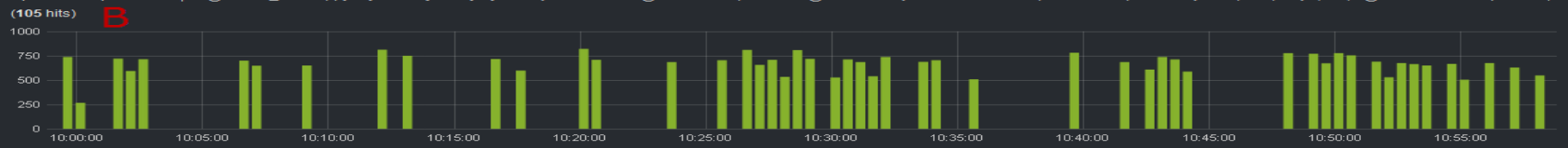
Query 0

Query

Graph 2

Avg Response Time - compositeNotify tnc Dequeue

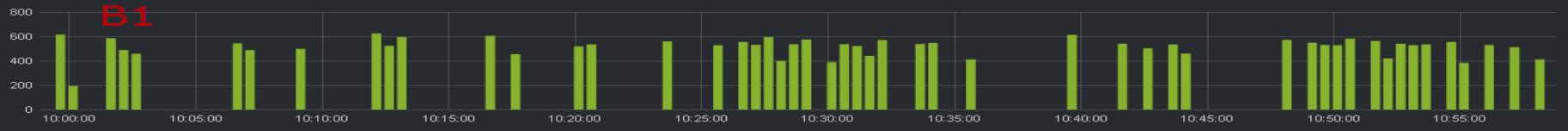
Zoom In Zoom Out | @source_host:/app[0-9]+\.ESB\jetson[14]\.coresys\.tmcs/ AND @fields.bam:"perf" AND @fields.activity:"com.ticketmaster.platform-template-notify-rest-post-proxy" (105) @fields.dur mean per **30s** | (**105** hits)



Graph 3

Avg Response Time - Template

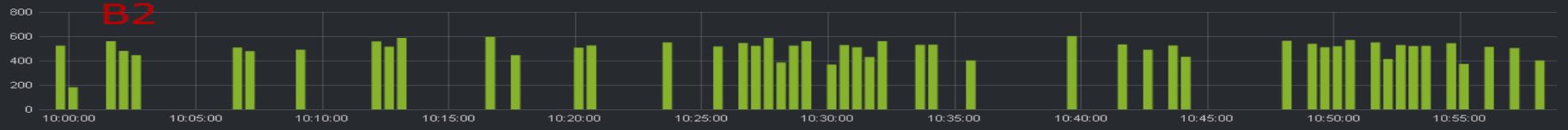
Zoom In Zoom Out | @source_host:/app[0-9]+\.TEMPLATE\jetson[14]\.coresys\.tmcs/ AND @fields.bam:"perf" AND @fields.activity:"com.ticketmaster.platform.template.web.service.RenderRestService.sendRenderRequest" (108) @fields.dur mean per **30s** | (**108** hits)



Graph 4

Avg Response Time - Template-->TCT

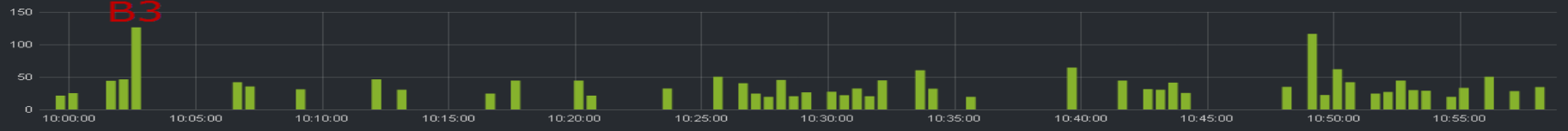
Zoom In Zoom Out | @source_host:/app[0-9]+.TEMPLATE.jetson[14].coresys.tmcs/ AND @fields.bam:"perf" AND @fields.activity:"TemplateClient.getTemplateResponse" (108) @fields.dur mean per **30s** | (**108** hits)



Graph 5

Average Response Time - Notify-Queuing

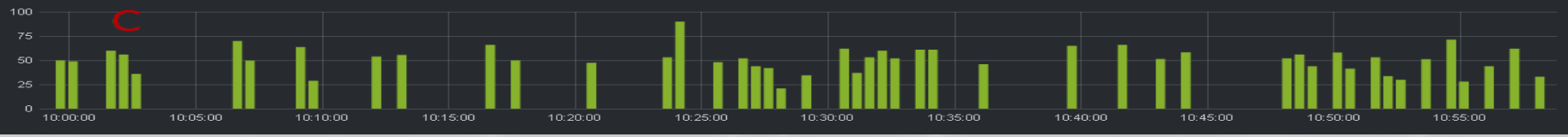
Zoom In Zoom Out | @source_host:/app[0-9]+\.NOTIFY\jetson[14]\.coresys\.tmcs/ AND @fields.bam:"perf" AND @fields.activity:"EmailRestService.sendAsyncEmail(..)" (86) @fields.dur mean per **30s** | (**86** hits)



Graph 6

Average Response Time - Notify-Dequeuing

Zoom In Zoom Out | @source_host:/app[0-9]+.NOTIFY.jetson[14].coresys.tmcs/ AND @fields.bam:"perf" AND @fields.activity:"EmailProvider.sendEmail(..)" (85) @fields.dur mean per **30s** | (**85** hits)



Monitoring Examples – END OF EXAMPLES

- We looked at performance dashboards for
  - a healthy test
  - a problematic test
  - production

1 FootnoteH
SOURCE: Source

Implementation

- How does this help your company improve performance in production?

1 FootnoteH
SOURCE: Source

How do you implement Performance MONITORING in your company?

- Require performance dashboards to qualify for production release. If resources are tight, the requirement could be relaxed, but the signaling would be clear.
- Create a reference set of performance dashboards. It is expected a single design would cover 90% of each application's need. Why? Because all applications have the same three key performance metrics.
- Offer the reference set of dashboards to all projects. Project owners could use them as is or modify them to fit their needs. Little burden means minimal resistance.

1 FootnoteH
SOURCE: Source

A generic performance dashboard.

How do you implement Performance ASSURANCE in your company?

- Prior to production release, the following performance requirements would need to be satisfied.
  - Pre-release Testing
    - Peak period scenario
    - Ramp up to first bottleneck test
  - Performance Monitoring dashboards

- Lets talk about Pre-release testing, the other arrow in our quiver.

1 FootnoteH
SOURCE: Source

How do you implement Performance ASSURANCE? in your company

- Prior to production release, the following performance requirements would need to be satisfied.
  - Pre-release Testing
    - Peak period scenario
      - Stakeholder provides SLA for peak production demand
      - Performance test simulates peak production demand while human users exercise the system to confirm acceptable user experience
      - PASS is given when SLA is met OR stakeholders accept results.
    - Ramp up to first bottleneck test
  - Performance Monitoring dashboards

1 FootnoteH
SOURCE: Source

How do you implement Performance ASSURANCE? in your company

- Prior to production release, the following performance requirements would need to be satisfied (or waived).
  - Pre-release Testing
    - Peak period scenario
    - Ramp up to first bottleneck test
      - Load is ramped up until the first significant bottleneck is found.
      - Useful for Ops to anticipate performance issues in production.
      - Provide information for the biz to create formal SLA's.
  - Performance Monitoring dashboards

1 FootnoteH
SOURCE: Source

How do you implement Performance ASSURANCE? in your company

- Prior to production release, the following performance requirements would need to be satisfied (or waived).
  - Pre-release Testing
    - Peak period scenario
    - Ramp up to first bottleneck test
  - Performance Monitoring dashboards
    - Performance monitoring is operational
    - Resource monitoring is operational
    - VM monitoring is operational
    - Dashboards available in both Pre-PROD and PROD environments

1 FootnoteH
SOURCE: Source

How do you implement Performance ASSURANCE? in your company

- Prior to production release, the following performance requirements would need to be satisfied (or waived).
  - Pre-release Testing
  - Performance Monitoring dashboards
    - Performance monitoring is operational
    - Resource monitoring is operational
    - VM monitoring is operational

    - Dashboards available in both Pre-PROD and PROD environments
      - Pre-PROD coverage provides feedback to performance problems
      - Pre-PROD coverage provides a testing ground for prod coverage.
      - Getting the right coverage will take practice.

1 FootnoteH
SOURCE: Source

How do pre-release testing and monitoring mitigate performance risk?

- We talked about two techniques of mitigating performance risk.
  - Pre-Release testing
  - Monitoring
- In practice, how does this theory map to performance problems?
- My experience says →

1 FootnoteH
SOURCE: Source

Sources of Production performance problems

Mitigation technique

Continuous s/w release – untested code
Shared Services – unpredictable demand
VM variability – starvation of resources

Production Monitoring

Pool misconfiguration
Poor DB indexing
Memory Leaks
Non-optimized code

Pre-Release Testing

Key
Performance defects/bugs

Questions

- Whew!! A lot of information

1 FootnoteH
SOURCE: Source

APPENDIX

1 FootnoteH

SOURCE: Source

Performance monitoring - Technologies

- Selecting an appropriate monitoring technology is highly dependant on your specific environment. Below I share the classes of monitoring technologies to consider for your solution.

Require customized application code to publish metrics, Log analyzer than employed
SysLog harvesting, log posting which include performance data points; (Kibana, Splunk)
Tcollector agents, performance information is pushed to a time series database (OpenTSDB)

Does not require customized application code: End-2-End vendor monitoring solutions	
Network sniffers	Network monitors or sniffer (OpNet)
Stitching	agent deployment required, piecing together transaction parts from header info (BlueStripe)
Transaction marking	agent deployment required, insert and than track headers
JVM monitors	agent deployment usually required (Dynatrace, AppDynamics)

1 FootnoteH

SOURCE: Source

Example of a performance log file used by Kibana to generate performance metrics:

- datetime="2013-12-12 11:59:59,538" severity="INFO " host="app6.template.jetcap1.coresys.tmcs" service_version="" client_host="10.72.4.75" client_version="" Correlation-ID="ab72d037-6362-11e3-80a4-f5667a7a5c6b" rid="" sid="" thread="Camel (337-camel-88) thread #42 - ServiceResolver" category="com.ticketmaster.platform.bam.strategies.PerformanceBAMStrategy" datetime="2013-12-12T11:59:59.538-08:00" bam="perf" dur="724" activity="template-call" camelhttppath="/template-notify-composite/rest/template-notify"

1 FootnoteH
SOURCE: Source

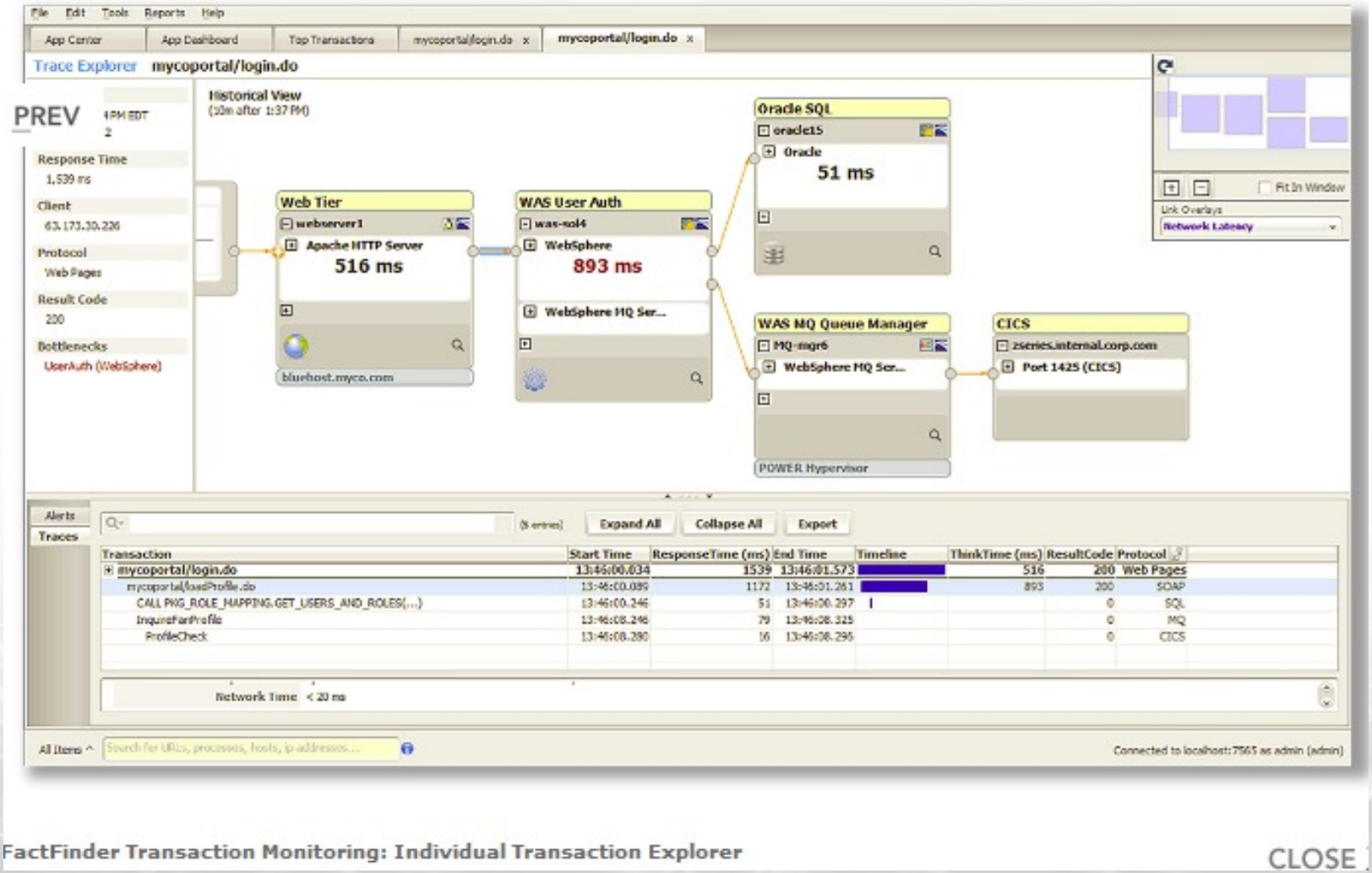
E2E Monitoring tool vendors:

- BlueStripe http://bluestripe.com/
- Op-Tier http://www.optier.com/
- AppDynamics http://www.appdynamics.com/
- Dynatrace
- http://www.compuware.com/application-performance-management/dynatrace-enterprise.html

- Gartner group does a nice evaluation of this tool space

1 FootnoteH
SOURCE: Source

E2E Transaction monitor example:



FactFinder Transaction Monitoring: Individual Transaction Explorer

CLOSE

1 FootnoteH

SOURCE: Source