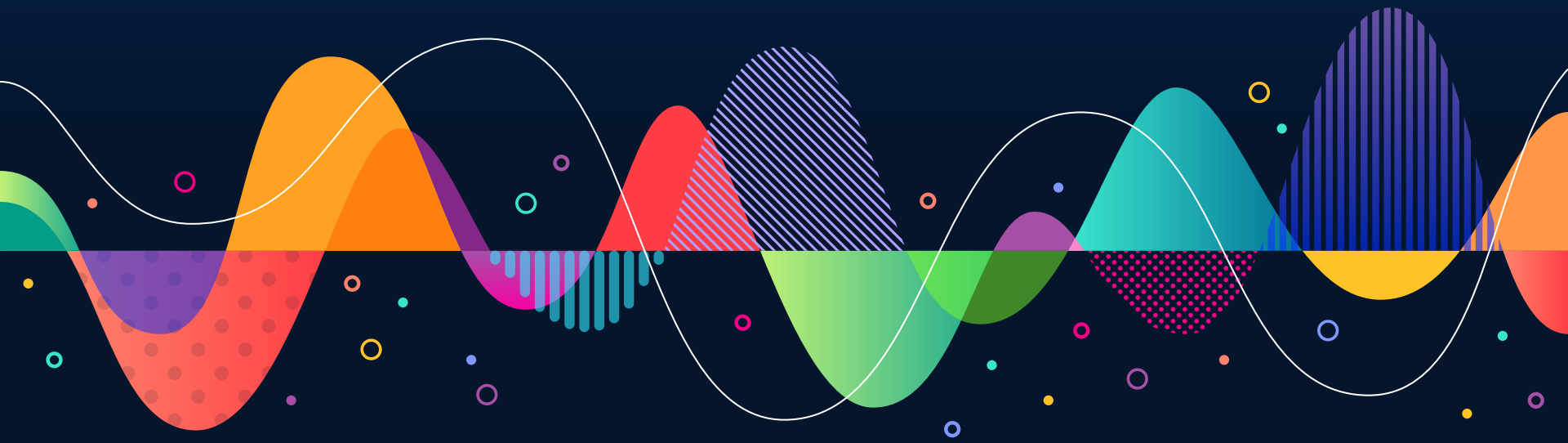
Performance Testing in Modern Contexts

Testing perspectives & perceptions
Cluj-Napoca
16 -17th of June 2022

Eric Proegler, USA

PERFORMANCE TESTING IN MODERN CONTEXTS



Hello! My name is Eric Proegler

Staff Test Infrastructure Engineer at Credit Karma in Oakland, California, USA

You can find me on Twitter at @3r1c_p

BEFORE 2020...

- Senior Director of Product Quality
- Product Manager for Load, Front End Monitoring, and Native Testing Tools
- Performance Testing Consultant
- Performance Lead/Test Lead/DBA for Enterprise Vendor, Developer was boring
- Also:
  - Past President, Association for Software Testing
  - Organizer, Workshop On Performance and Reliability
  - Currently (as in, at this moment) speaking at my ~40th Conference

WHAT ABOUT YOU?

Who are you?

Name?

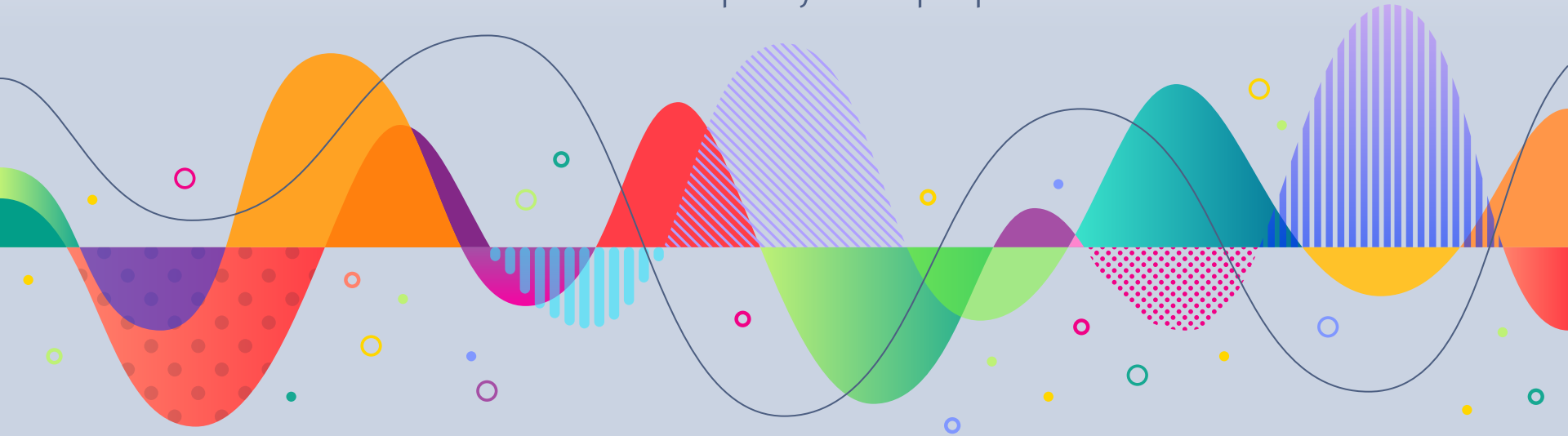
Work?

Experience with Performance?

What would you like to learn about Performance?

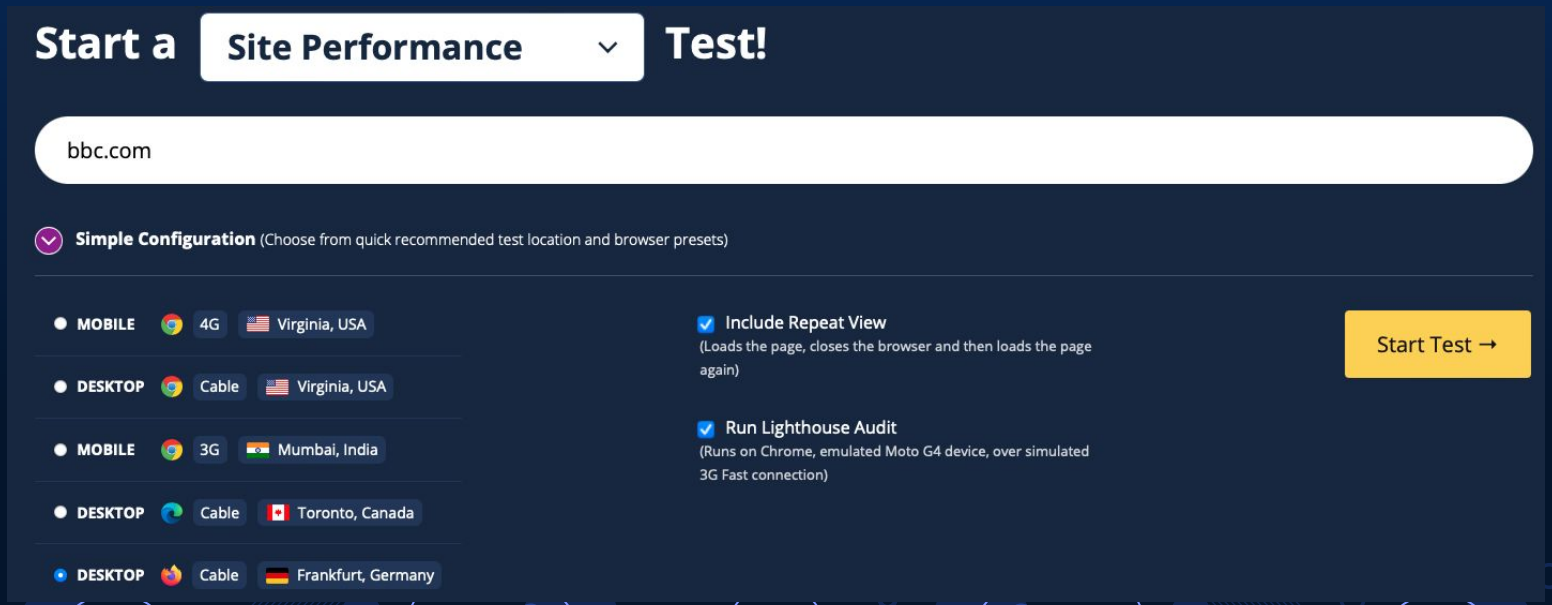
LET'S PERFORMANCE TEST!

Please open your laptop



PLEASE OPEN A BROWSER TAB TO https://www.webpagetest.org

LET'S RUN A TEST



YOU ARE NOW A PERFORMANCE TESTER

Lots of Things We *COULD* look at. Not a Science! Start with:

- Time until Page Looks/Feels Loaded - ***Response Time***
- Improvement on 2nd/3rd Visit - ***Caching Strategy***
- Total Bytes - ***Bandwidth Requirements***
- Blocking and Client CPU - ***Front End Efficiency***
- Waterfall Charts - ***End to End Performance***

TAKE A FEW MORE MINUTES

We’re coming back to this a little later

- Pick something in your results you want to know more about
- Don’t close your browser tab(s) yet!

MORNING: FRONT END, REQUIREMENTS

9:45 - 10:10 (25)	Activity 1: Web Page Test
10:10 - 10:30 (20)	Performance as a Requirement
10:30 - 10:45 (15)	Front End Theory - Browsers, Native Apps, and Failure Modes
10:45 - 11:00 (15)	Break
11:00 - 11:20 (20)	Activity 2: More Web Page Test - What else can we learn?
11:20 - 11:45 (25)	Activity 3: Chrome Dev Tools
11:45 - 12:00 (15)	Activity 4: Lighthouse, then Lunch!

AFTERNOON: LOAD, BACK END, REPORTING

1:20 - 1:50 (30)	Activity 5: JMeter. https://github.com/ericproegler/romania-testing/ setup
1:50 - 2:15 (25)	A Lot About Load Testing
2:15 - 2:30 (15)	Observability and Logging
2:30 - 2:45 (15)	Break
2:45 - 3:15 (30)	Back End Theory: Clouds, Architecture, Parallelism, and Queues
3:15 - 3:30 (15)	Reporting on Performance
3:30 - ? (?)	Q&A / Anything we want to revisit?

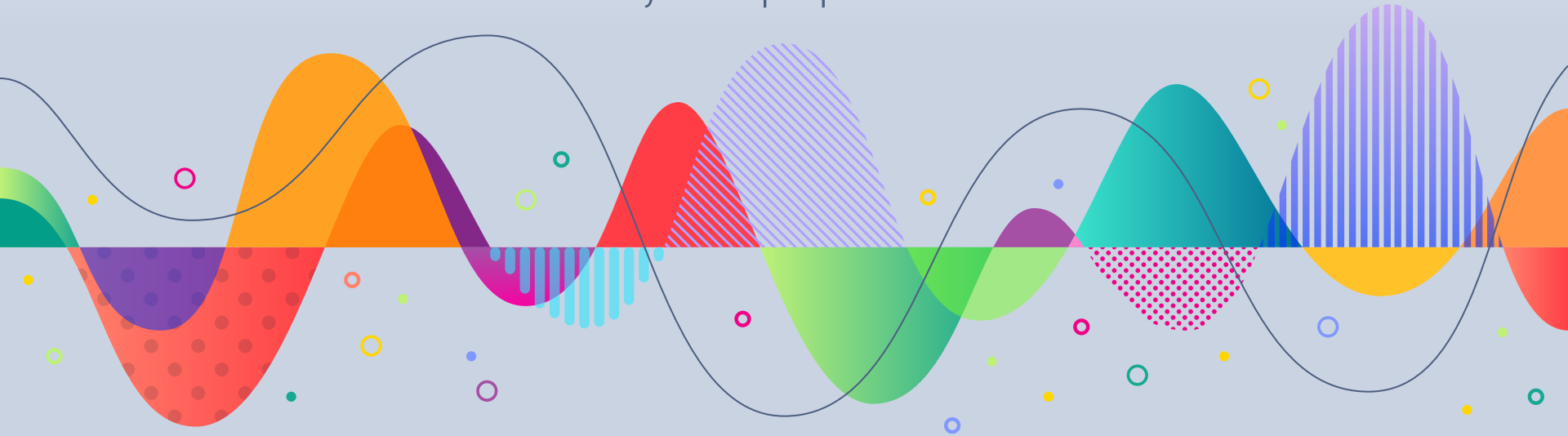
LET'S MAKE YOUR LAPTOP A LOAD RIG

Hopefully everyone can get this working by lunch...

1. https://github.com/ericproegler/romania-testing/
2. Download Java 8 (~ 85mb) and install it
3. Download JMeter (~ 74mb) and extract it
4. Download Script (< 1mb) and move it to /bin/

PERFORMANCE AS A REQUIREMENT

Please close your laptop for a few minutes



SOME REQUIREMENTS ARE TACIT

NFRs are Non-Functional Requirements. They need your help!

- Security: “Meets Anti-Gremlin Standard 2.3” ***Which includes?***
- Reliability: “99.9% Uptime” ***24x7x365? Business Hours? Which functionalities failing counts as Downtime?***
- Meets Regulatory Data Governance Standards ***In which countries? For which data? How do we know for sure?***

PERFORMANCE AS ESSENTIAL REQUIREMENT

Slow Performance can make a system unacceptable/unfit for purpose

- Once Users think software is Slow, hard to see anything different
- ...But Users know Slow when they see it
- In e-commerce, tenths of seconds can be costed
- Workers' time is always expensive. Distractions can be worse
- In Enterprise, unhappy users leads to system replacements

PERFORMANCE REQUIREMENT: RESPONSE

"Make Sure the System is (Performant/Responsive/Fast Enough)!"

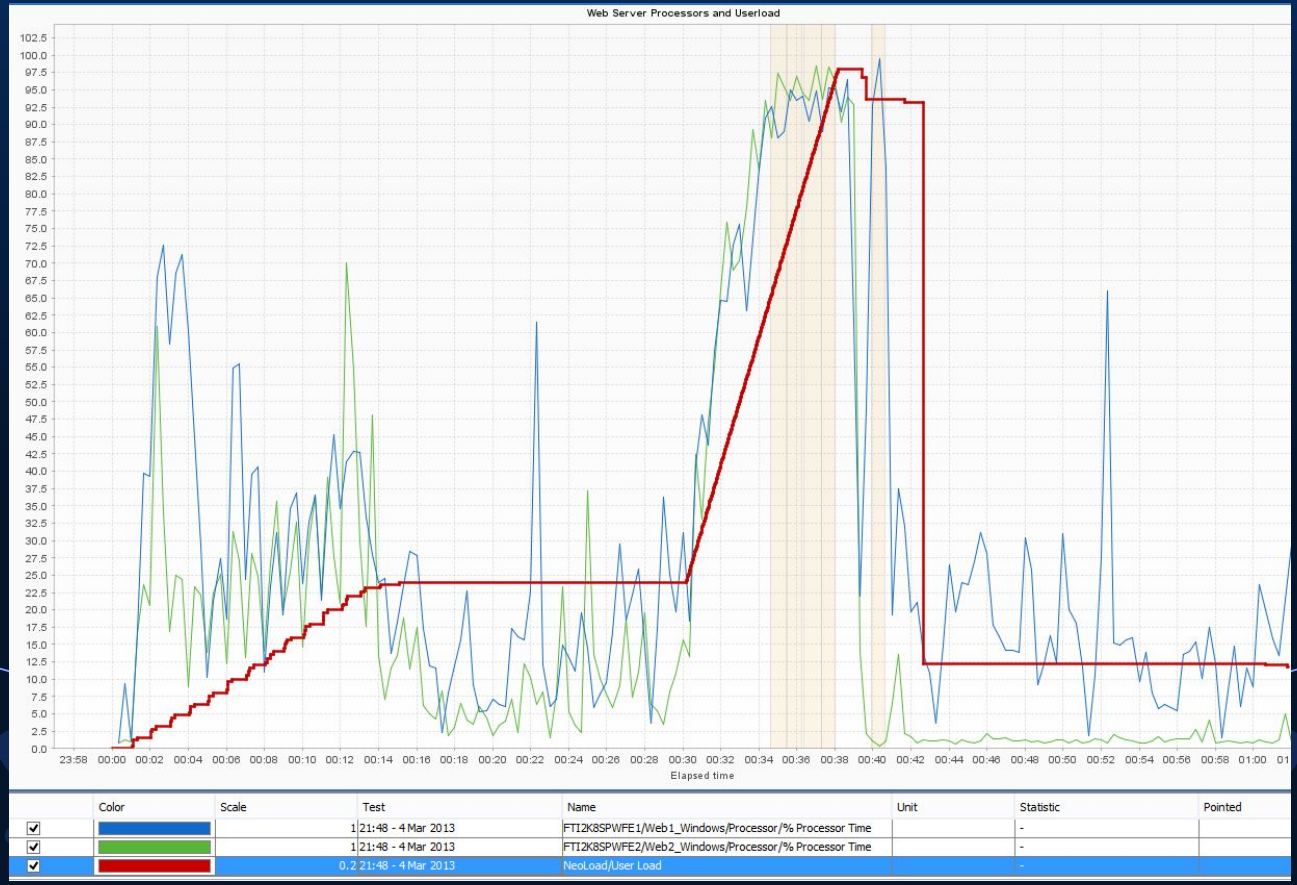
- How fast is fast enough? - ***Response Time***
- "Whatever the Industry Standard is"
  - If you're Amazon, under 1 second. E-Commerce 2/3 seconds
  - 5 seconds is an eyeroll. 10 seconds is a sigh. 15 seconds is a bye
  - Not entirely in our control (Last Mile, Device) - more on this later

PERFORMANCE REQUIREMENT: SCALE

"Make Sure the System Scales!"

- How many users? - ***Peak Load***
  - Usually Busiest Hour of Busiest Day, May Not be Known
- At what arrival rate? - ***Point Load***
  - Eurovision/Product Drop/Viral Moment, Usually Unknown
- With what responsiveness? - ***Performance SLA***

MORE ON ARRIVAL RATES



PERFORMANCE RISKS

- Scalability - System activities are expensive/inefficient; implies it won't support larger workloads/users
- Capacity - System has architectural limits
- Concurrency - System has race conditions or blocking issues that can crash/delay/block/error the system or a user
- Reliability - System should be able to run for extended periods of time

RISK: SCALABILITY

Expensive operations mean systems won't scale well

- Won't get faster with load!
- Ops Problem? Solve with hardware (more pods)?
- Tall Stacks -> Wirth's Law: "Software is getting slower more rapidly than hardware gets faster"
- Subject to use patterns and user models

RISK: SCALABILITY

What Does a Problem Look Like?

- Longer response times is a clue
- “High” CPU/Memory/Storage/Network Utilization

RISK: CAPACITY

System can't support the expected load structurally/as engineered

What Does a Problem Look Like?

- Response time very sensitive to load
- Hard or Soft Resource Limitations
- CPU/Network Limitation
- Increasing I/O Latency
- Database threads and other queues

RISK: CONCURRENCY

Operations that contend and collide (Race conditions, database locks, contention points)

What Does a Problem Look Like?

- Infrequent functional issues that seem to only occur under load
- Process crashes
- Not easily reproducible

RISK: RELIABILITY

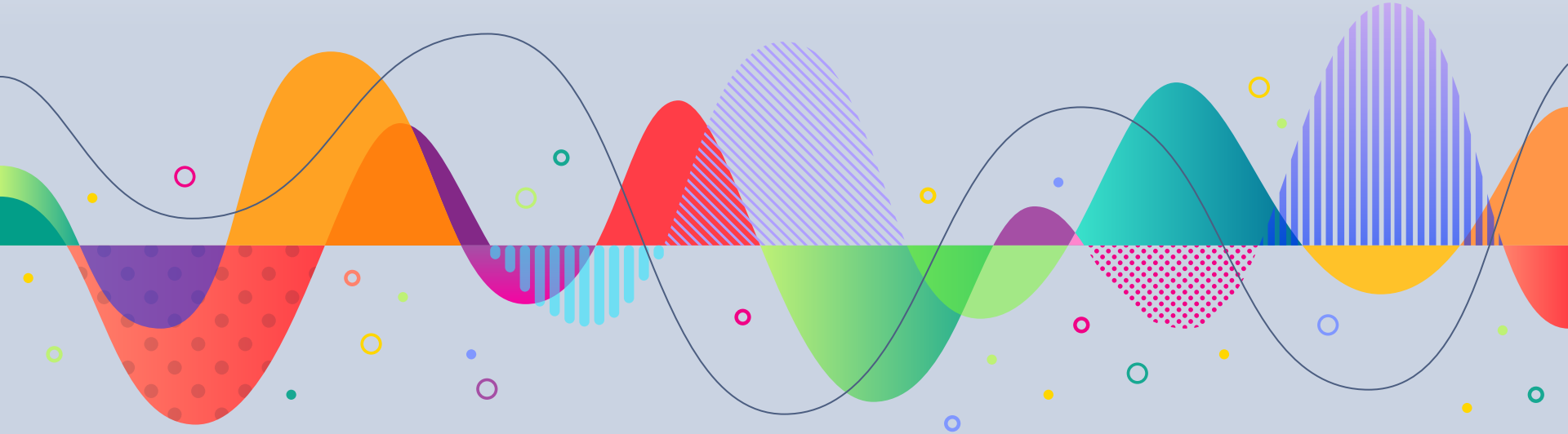
Degradation over time. System becomes slower, less predictable, or eventually fails.

What Does a Problem Look Like?

- Memory or Object Leaks (counts climbing)
- More frequent Garbage Collection
- Decaying response times

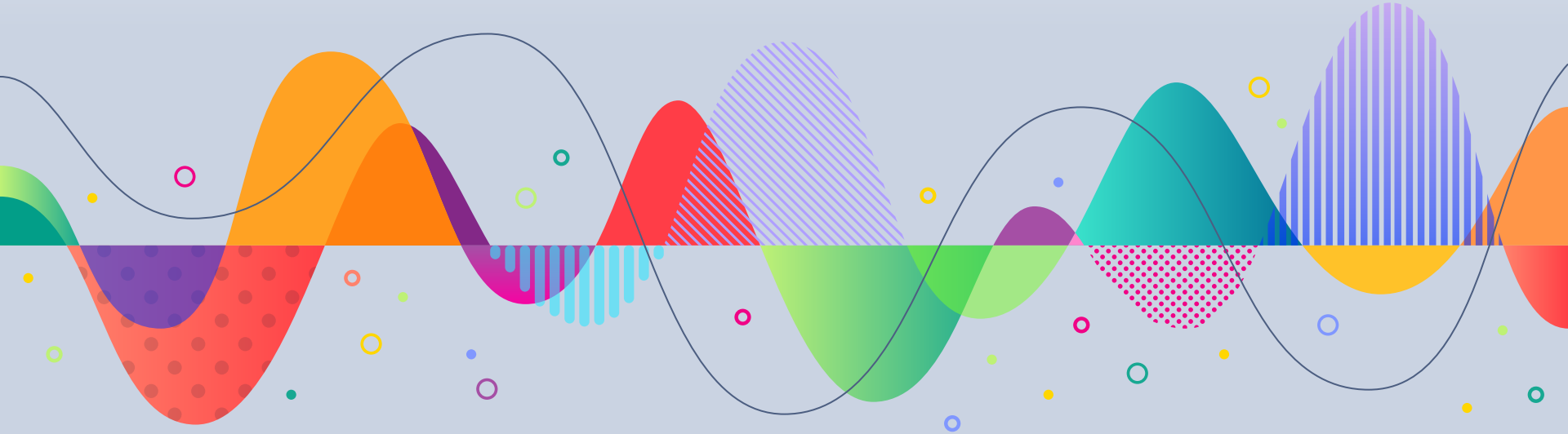
BREAK TIME

LET'S REFRESH THE COOKIES!



FRONT END THEORY

What's the Scenario?



WHERE MOST RESPONSE TIME IS SPENT

Compute moved to edge - that is the glowing misery rectangle, either landscape or portrait, in front of you

- Servers (and CDNs) send code and files
- Network provides medium for transfer
- Client does most of the work processing code and files, then rendering the DOM

CLIENTS RENDERING A WEB PAGE

All of these things are happening on the Device

- Requests for Resources (Text, Images, Code, Resource Files)
- Parsing Text: HTML/CSS/JSON/XML
- Executing Scripts (Almost always JS, was more Java in the past)
- Drawing (and redrawing) the screen

RENDERING A WEB PAGE: REQUESTS

Request can be for a specific resource or directory (site root)

- Could be CDN (Content Delivery Network) or 3rd Party
- DNS first - resolve Target of the request (IP Address)
- Establish Connection to Target
- Send Request to Target, get Response/Redirect/Error
- Read Response - based on content, might make more requests

RENDERING A WEB PAGE: RESPONSES

You might already know. But a quick refresher of key points:

- Some things (302 Redirect) are apparent to the Client. Others (Load Balancing, Request Processing) are not
- Transmission of Data subject to Bandwidth for whole Client path
- Many requests means Latency is expensive
- Some responses are blocking

RENDERING A WEB PAGE: PARSING

- HTML document is read, resources are all requested
- CSS Rules have to be processed. This can be done inefficiently
- Some data arrives in JSON objects - sometimes these are massive in ways developers do do not understand
- Frameworks like React defer some traffic/parsing to avoid blocking, but it still is a lot of data to process

RENDERING A WEB PAGE: EXECUTING CODE

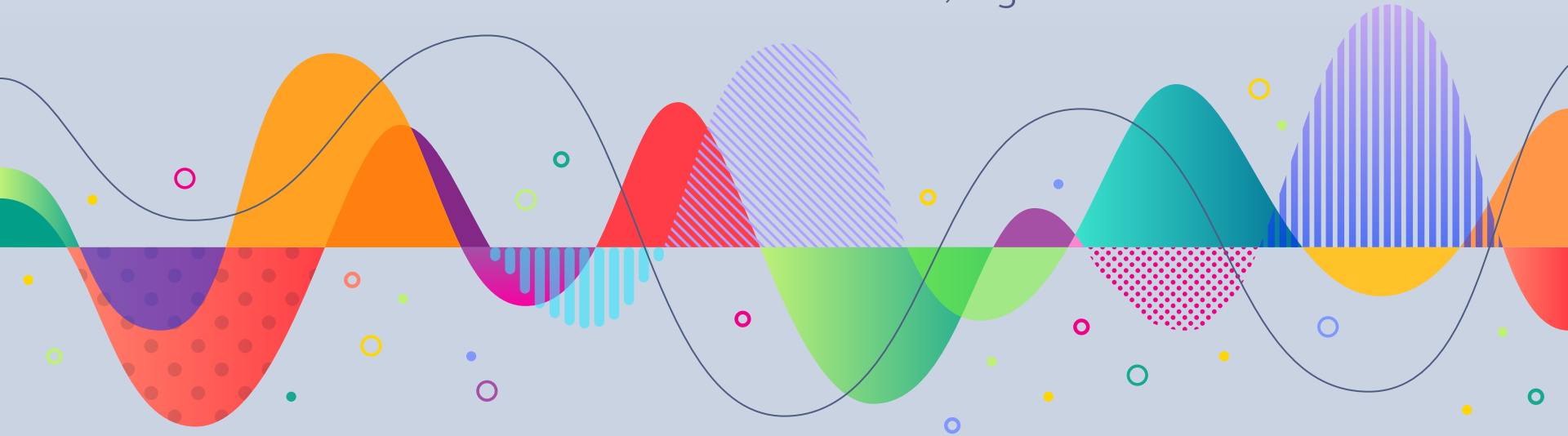
- JS snippets for tracking and monitoring have to be interpreted and executed on the fly
- Another place where blocking can happen
- Since many files are only referenced/abstracted (3rd Party), subject to unexpected changes that slow the whole page down

RENDERING A WEB PAGE: INTERACTIVE

- Can keep processing in the background
- When can the user see it?
- When can the user interact with it? Business users may be both familiar and impatient
- This is not an exact science

BACK TO PERFORMANCE TEST RESULTS

Saved Your Browser Tabs, Right?



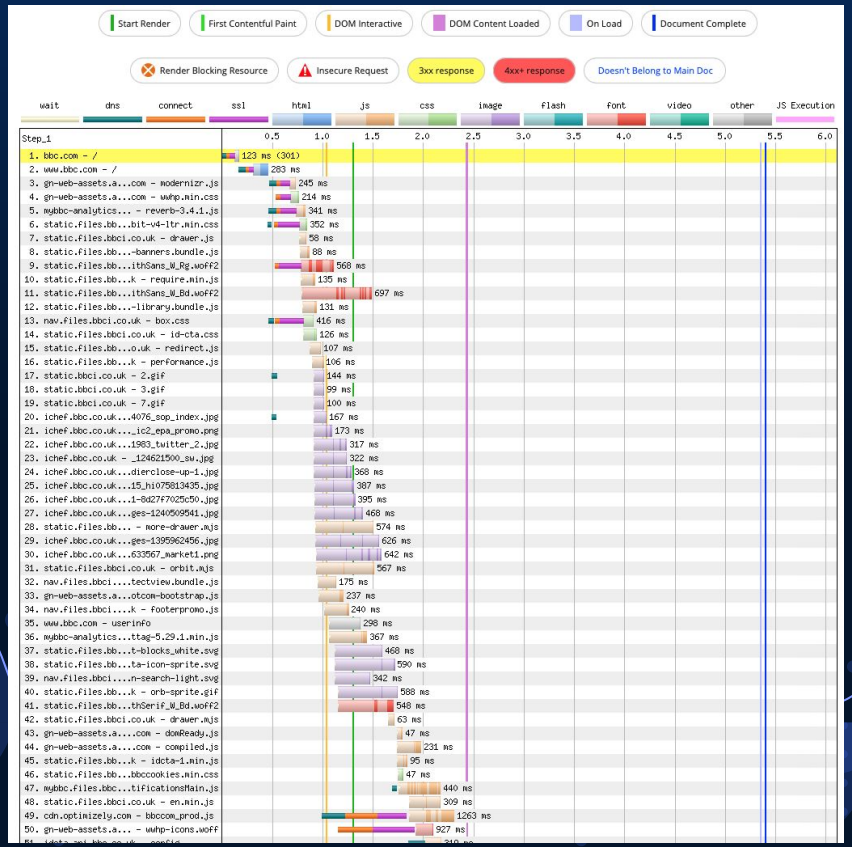
LET'S POKE AROUND A LITTLE MORE

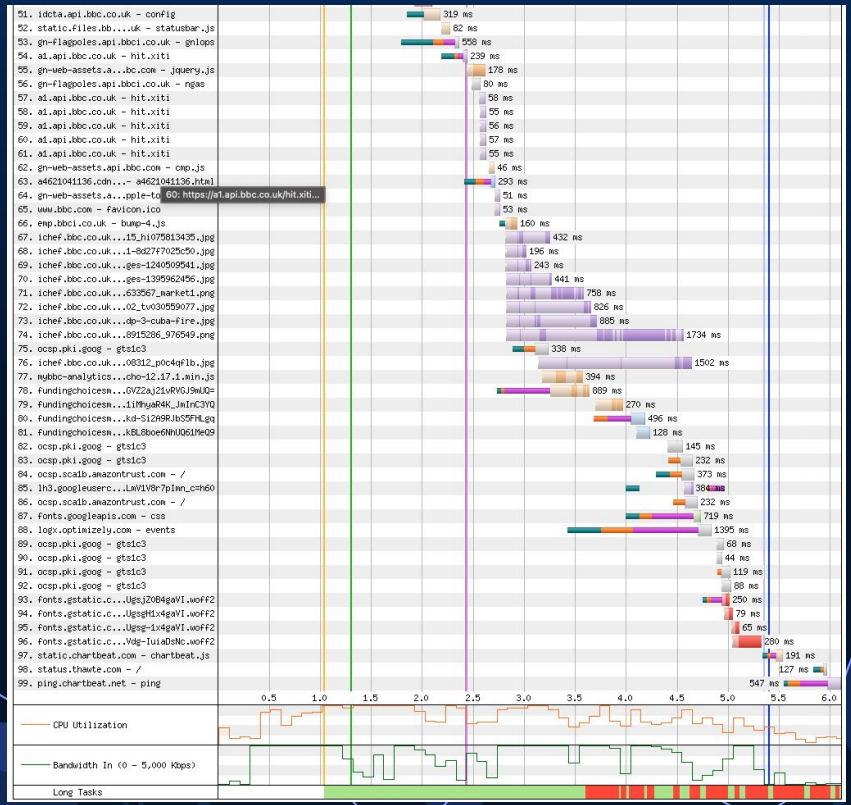
Remember our Test Results? Let's look at those a little more.

- What do you want to know more about?
- If you closed browser tabs - just run another test real quick
- We could spend a lot of time here - I'm aiming for about 15 minutes


North America
Virginia - EC2
Salt Lake City, Utah - GCE
California - EC2
Toronto, Canada - EC2
South America
Sao Paulo, Brazil - EC2
Europe
Ireland - EC2
London, UK - EC2
Paris - EC2
Amsterdam, NL - GCE
Frankfurt, Germany - EC2
Milan, Italy - EC2
Stockholm, Sweden - EC2
Africa
Cape Town, South Africa - EC2
Middle East
Bahrain - EC2
Dubai, UAE - Azure
Asia
Mumbai, India - EC2
Bangkok, Thailand - Tencent
Singapore - EC2
Jakarta, Indonesia - GCE
Hong Kong, China - EC2
Shanghai, China - Tencent
Beijing, China - EC2
Beijing, China - Tencent
Ningxia, China - EC2
Taiwan - GCE
Seoul, Korea - EC2
Tokyo, Japan - EC2
Osaka, Japan - EC2
Oceania
Sydney, Australia - EC2
Melbourne, Australia - Azure
Physical Devices
Los Angeles, CA - M1 Mac Mini/iOS (Chrome,Firefox,Safari)
New York, NY - M1 Mac Mini/iOS (Chrome,Firefox,Safari)
New York, NY - Android Devices
Dulles, VA - Android Devices
Cluj-Napoca
Custom headers
Add custom headers to all network requests emitted from the browser
Inject Script
JavaScript to run after the document
Start a
Site Performance
Test!
http://bbc.uk
Simple Configu
MOBILE
Core Web Vitals
Lighthouse
Visual Comparison
Traceroute
37
Cable (5/1 Mbps 28ms RTT)
DSL (1.5 Mbps/384 Kbps 50ms RTT)
3G Slow (400 Kbps, 400ms RTT)
3G (1.6 Mbps/768 Kbps 300ms RTT)
3G Fast (1.6 Mbps/768 Kbps 150ms RTT)
4G (9 Mbps, 170ms RTT)
LTE (12 Mbps, 70ms RTT)
Mobile Edge (240 Kbps/200 Kbps 840ms RTT)
2G (280 Kbps/256 Kbps 800ms RTT)
56K Dial-Up (49/30 Kbps 120ms RTT)
FIOS (20/5 Mbps 4ms RTT)
Native Connection (No Traffic Shaping)
Custom
Desktop
Chrome
Chrome Beta
Chrome Canary
Firefox
Firefox Nightly
Firefox ESR
Brave
Edge
Chrome Device Emulation
Motorola G (gen 4)
Nexus 5
Nexus 5X
Google Pixel
Google Pixel XL
Google Pixel 2
Google Pixel 2 XL
Motorola G (gen 1)
Samsung Galaxy S5
Samsung Galaxy S7
Samsung Galaxy S8/S8+/Note 8
Motorola E
Android One
Nexus 7
Nexus 7 - Landscape
iPhone 5C
iPhone 6/7/8
iPhone 6+/7+/8+
iPhone X
iPad
iPad Mini

CHASE WATERFALLS YOU'RE NOT USED TO

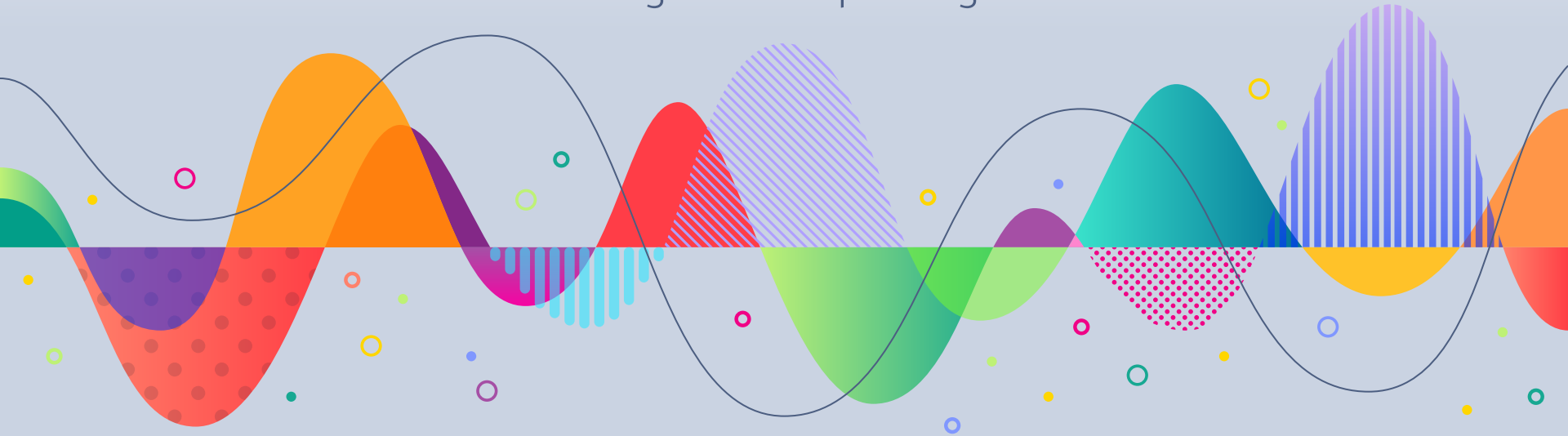




5.

BROWSER DEVELOPER TOOLS

Understanding and Manipulating The DOM



BROWSER DEV TOOLS

You may already know. If not, you are in for a treat!

- Every browser - menu item, key chord, or Inspect
- Source, Console, Request/Response, and more
- Check for errors, run scripts, etc - we'll just look at Perf today
- Let's Poke Around!

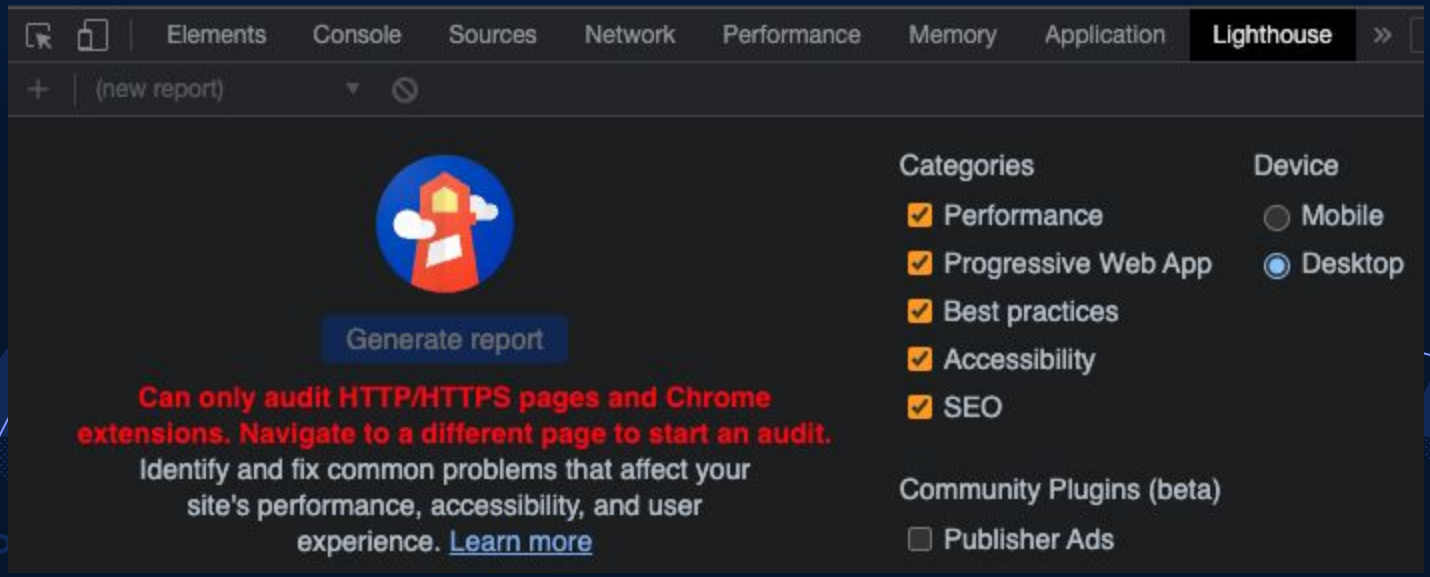
6.

LIGHTHOUSE

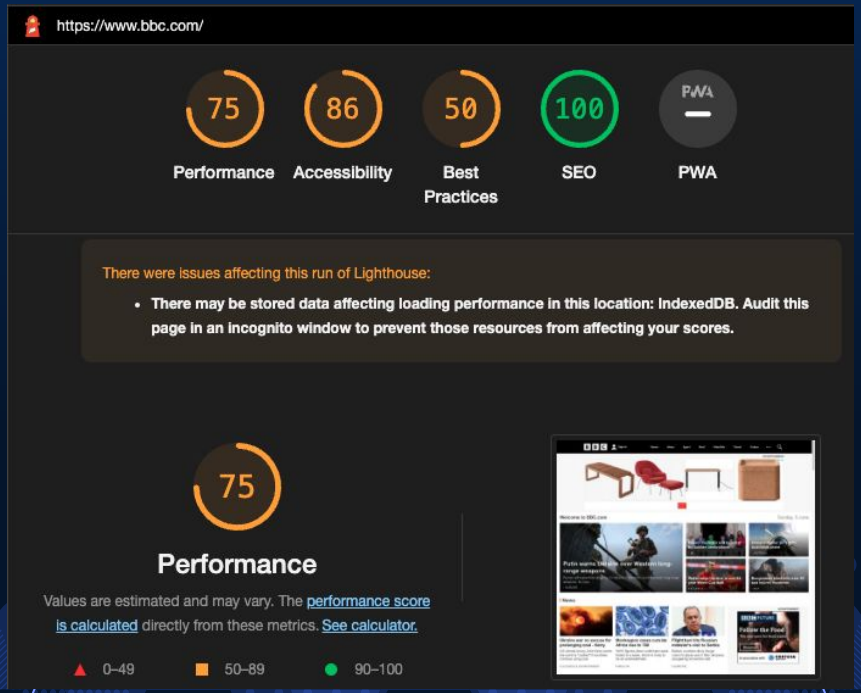
Staying Off the Rocks

TIME TO GET AN EXPERT CONSULT

- Please Open a Chrome Incognito Window
- Don't Enter a URL Yet - and Open Dev Tools
- Pick Lighthouse Tab, and Choose Desktop. Then Generate!

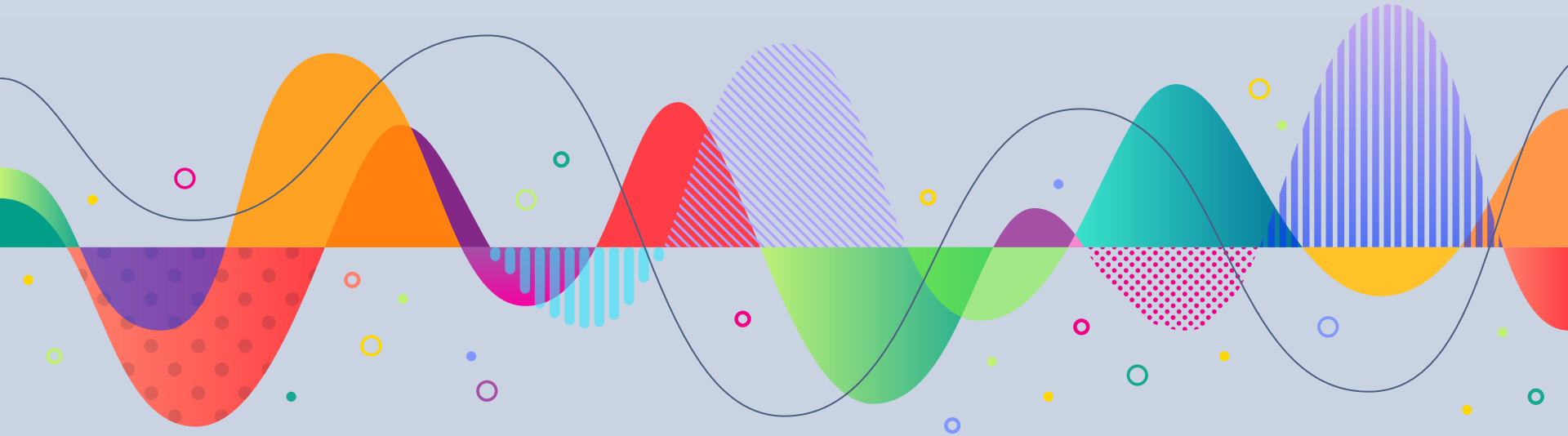


LET'S AUDIT


https://www.bbc.com/
75
Performance
86
Accessibility
50
Best Practices
100
SEO
PWA
PWA
There were issues affecting this run of Lighthouse:
There may be stored data affecting loading performance in this location: IndexedDB. Audit this page in an incognito window to prevent those resources from affecting your scores.
75
Performance
Values are estimated and may vary. The performance score is calculated directly from these metrics. See calculator.
0–49
50–89
90–100

LUNCH

LET'S REFRESH THE COOKIES!



HOW WAS LUNCH?

Any thoughts from what we covered this morning?

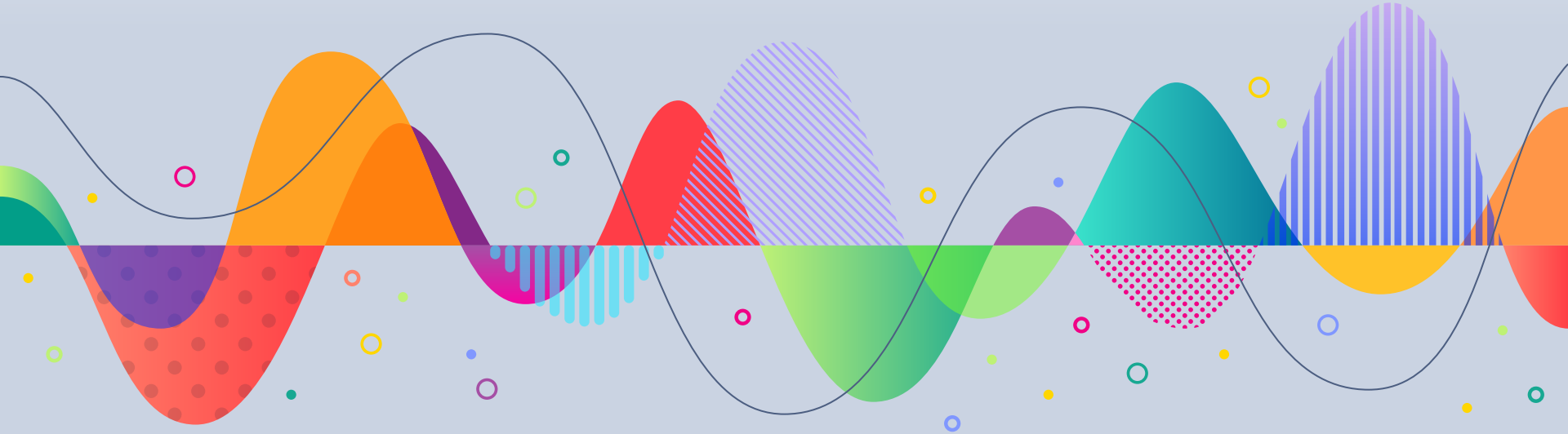
Anything you want to hear more about, or want to get to?

AFTERNOON: LOAD, BACK END, REPORTING

1:20 - 1:50 (30)	Activity 5: JMeter. https://github.com/ericproegler/romania-testing/ setup
1:50 - 2:15 (25)	A Lot About Load Testing
2:15 - 2:30 (15)	Observability and Logging
2:30 - 2:45 (15)	Break
2:45 - 3:15 (30)	Back End Theory: Clouds, Architecture, Parallelism, and Queues
3:15 - 3:30 (15)	Reporting on Performance
3:30 - ? (?)	Q&A / Anything we want to revisit?

JMETER

Kill it With Fire



LET'S MAKE YOUR LAPTOP A LOAD RIG

Hopefully everyone can get this working. Sorry in advance...

1. https://github.com/ericproegler/romania-testing/
2. Download Java 8 (~ 85mb) and install it
3. Download JMeter (~ 74mb) and extract it
4. Download Script Files (< 1mb) and move them to /bin/

LET'S MAKE YOUR LAPTOP A LOAD RIG

Hopefully everyone can get this working. Sorry in advance...

5. Launch apache-jmeter-5.4.3/bin/ApacheJMeter.jar
6. File -> Open -> BlazeDemo.jmx
7. Push Start (Courage, Patience, Curiosity, and Resolve)

YOU ARE NOW A LOAD TESTER!

Let's look at our script for a moment...

- Requests - What Page We Request
- Test Data - Parameterization in Search Flights
- Post Parameters - Select/Confirm Flight
- Timers - Random and Static
- Assertion - Check for Thank You

FURTHER EXPERIMENTS

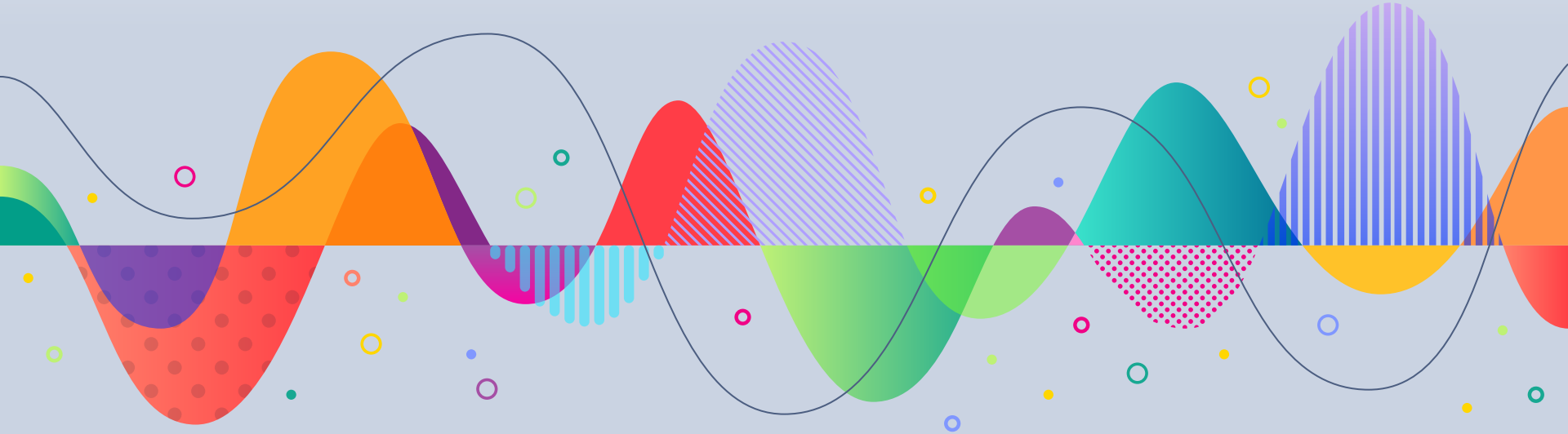
Try some Tweaks, but remember...

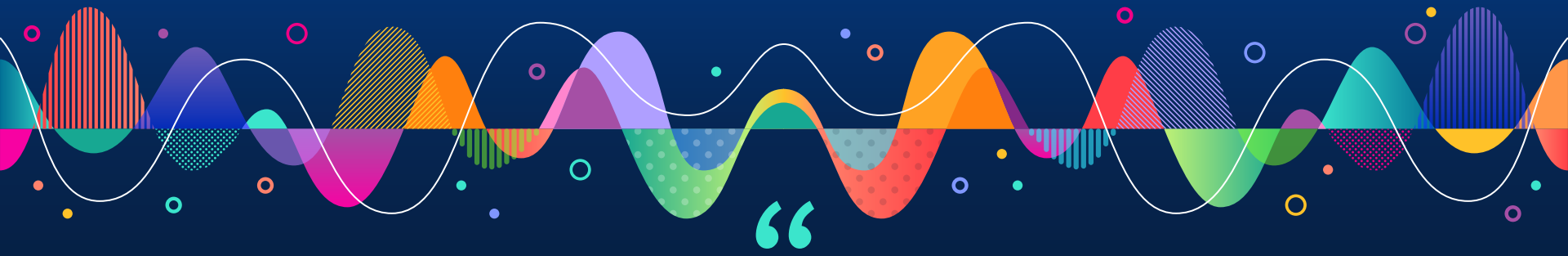
1. It's easy to take a test site down - be cautious
2. You can always hit that 

3. Site is blazedemo.com if you want to add some asserts?
4. What did you learn?

LOAD TESTING

DDOS for future QOS





“Load Testing:

The Coolest Video Game Ever Invented...

Though the Graphics Kinda Suck”

\- Michael Pearl

LOAD TESTING, AN ARCANE ACTIVITY

- Expensive (Time): create tests, design workload, monitoring setup, results analysis, reporting, remediation, etc
- Expensive (Delay): Might not really start until code is almost complete, then a gate to release. Someone will be unhappy
- Expensive ($/€/L): tool license, injector hours, salaries, etc)
- Still could be cheaper in the long run than a failing application

LOAD TESTING: IF YOU MUST

Load Testing Often Purports to Answer the Following:

Will the completed, deployed system support

(*a*, *b*...) users performing (*e*, *f*...) activities *(Scripts),*
at (*j*, *k*...) rates *(Load Model),*
on *mn*... configuration *(Environment),*
under *rs*... external conditions *(LOL),*
meeting *x*, *y*... response time goals? *(Requirements)*

LOAD TESTS ARE EXPERIMENTS

- Conditions - Create Load, Stress the System
  - Scripts and Workload Model
  - Execution Environment/System Under Test
- Observations - While Under Load, Examine the System
  - Response Times against Load Increases
  - Resource Measurements (when available)

REQUIREMENTS BEING TESTED

- Does the System Scale as Expected?
- Is Response Time Fast Enough? What is Fast Enough?
- Is Capacity Enough? What is Enough? Per Pod?
- Is the System Reliable Enough? Over How Long?
- The Most Interesting Testing is for Learning

SCRIPTS

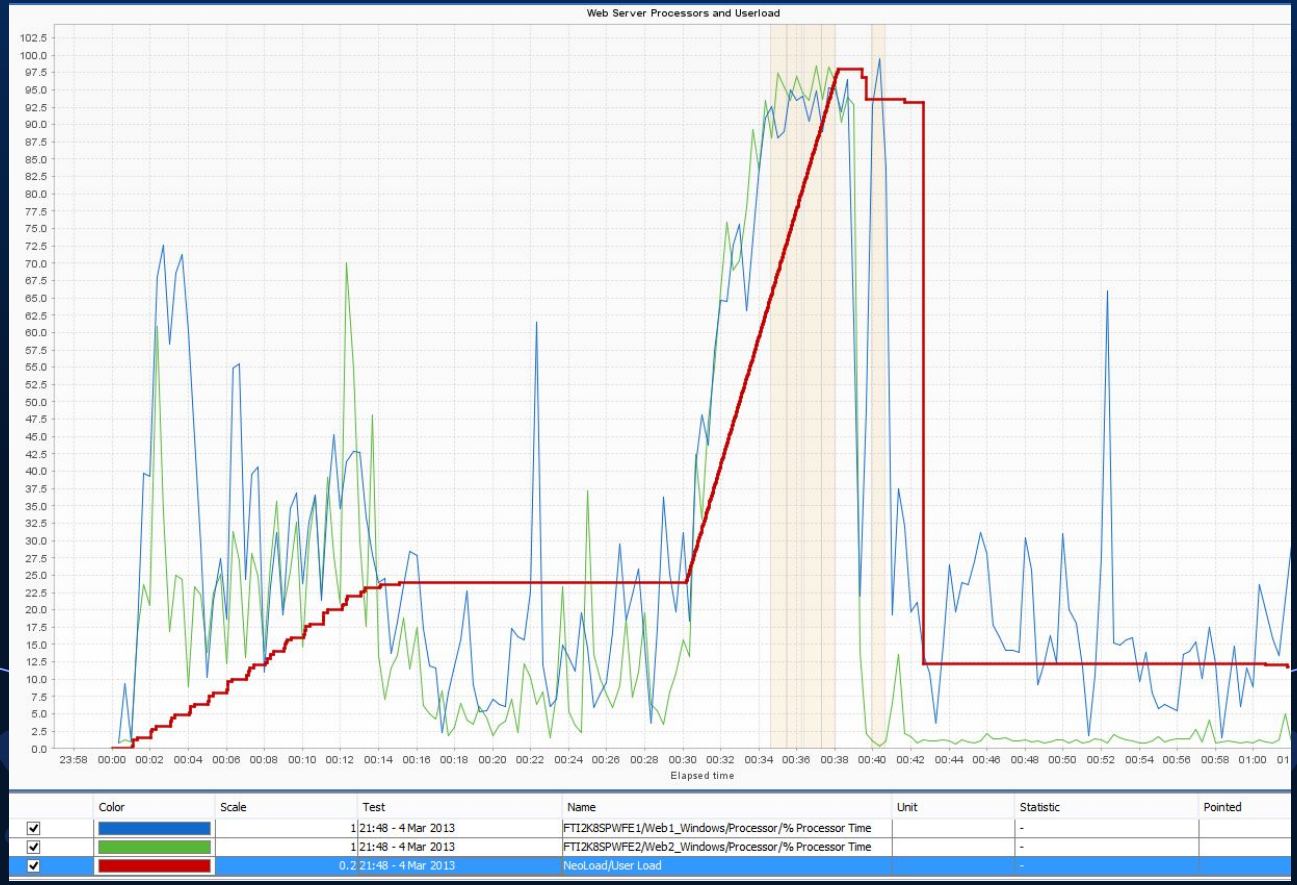
Load Tests typically replay 1-10 Scripts

- No Client - recreate HTTP requests to simulate many clients
- Parameterize for Accounts/Test Data
- Login Patterns like Tokens can be complex
- Use Timers to Simulate Delays - Think Time
- Load Scripts are like Automation, only more so (brittle, etc)

LOAD MODEL

- Reflect today + x%? What about new apps?
- Explicitly consists of Users, Scripts, Ramp rates, Duration
- Implicitly includes think time (length of script execution)
  - “Random” makes homogenous workloads
- Can sometimes be adjusted during runs

REMEMBER SHARKY?



	Color	Scale	Test	Name	Unit	Statistic	Pointed
☑		1	21:48 - 4 Mar 2013	FTI2K8SPWFE1/Web1_Windows/Processor/% Processor Time		-	
☑		1	21:48 - 4 Mar 2013	FTI2K8SPWFE2/Web2_Windows/Processor/% Processor Time		-	
☑		0.2	21:48 - 4 Mar 2013	NeoLoad/User Load		-	

ENVIRONMENT & MONITORING

- “Prod” or Test? “Prod” is almost never Prod, unless it is Prod.
  - Can run Prod + Load, if you have confidence and resolve
- When Servers were Hardware, this all mattered a lot more
- You can still learn a lot from Test. Or Local
- Resources (CPU, Mem Mgmt, I/O, DB, etc) is another discipline
- Auto-Scaling is like Auto-Driving - not really, and dumb

EXTERNAL CONDITIONS

- System's External Conditions:
  - Noisy Neighbors - w/Virtualization/Cloud, more difficult
- Client Conditions
  - Running a real browser w/rendering and JS, etc
  - Background/competing tasks
- **Last Mile Network Conditions**

SOFT REQUIREMENTS

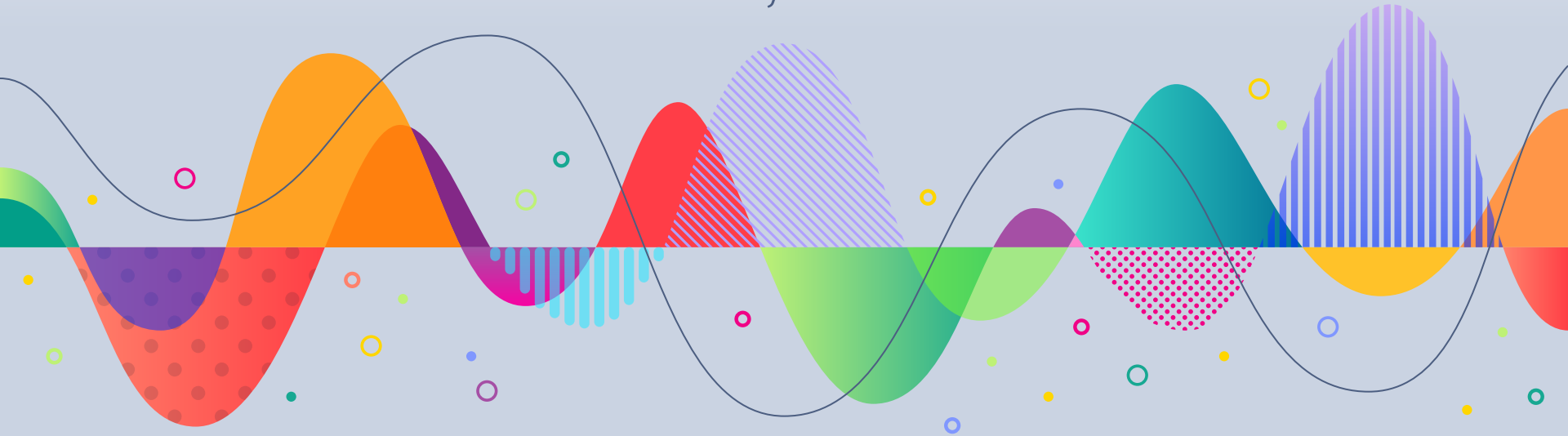
- Well-Defined and Well-Reasoned plan to share
- Secured Executive Sponsorship: Whose Name Can You Drop?
- Permission to Blast Away; Coordination with Teams responsible for resources
- Credibility to reschedule if scripting is getting behind, rerun tests if something goes wrong, and to quickly tune

INSTEAD OF LOAD TESTING...

- Observe Production: Real Users, Real Activities and Levels
- Log on to Prod
  - When is a busy time? Go see how your app performs then. Is it slower than other times?
  - Use the Front End techniques we've discussed...
- Modern Canary/Blue-Green/Progressive - what's the diff?

OBSERVABILITY AND LOGGING

That All Sounds Good in Theory - How Does it Work in Practice?



EVERYONE TESTS IN PROD...

...Some do it on purpose

- The Only “Real”, Where it Matters
- Rich source of information - lots of instrumentation already
- Canary/Blue-Green are pretty standard
- Progressive Deployments are next

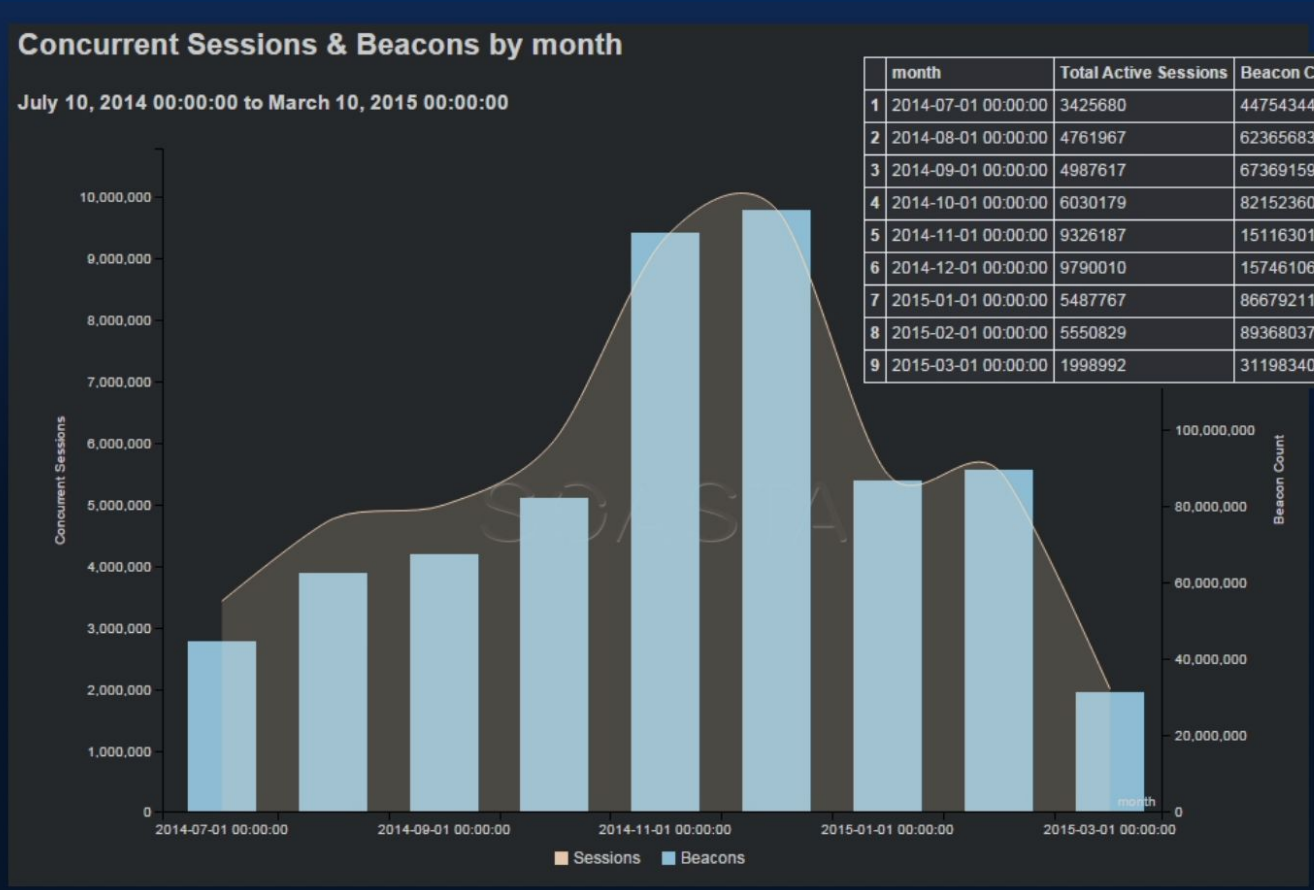
OBSERVING PRODUCTION

- **“Observability”** - New Relic, Dynatrace, AppDynamics, mPulse
  - Metrics, Aggregated: Resources and Dashboards
- **~ Observability** - Splunk, Datadog
  - Logs, Context: Finding Specific Events and Tracing Them
- **Observability** - Honeycomb.io
  - Telemetry: Metrics, Logs, Traces, Structured Events

ASKING QUESTIONS

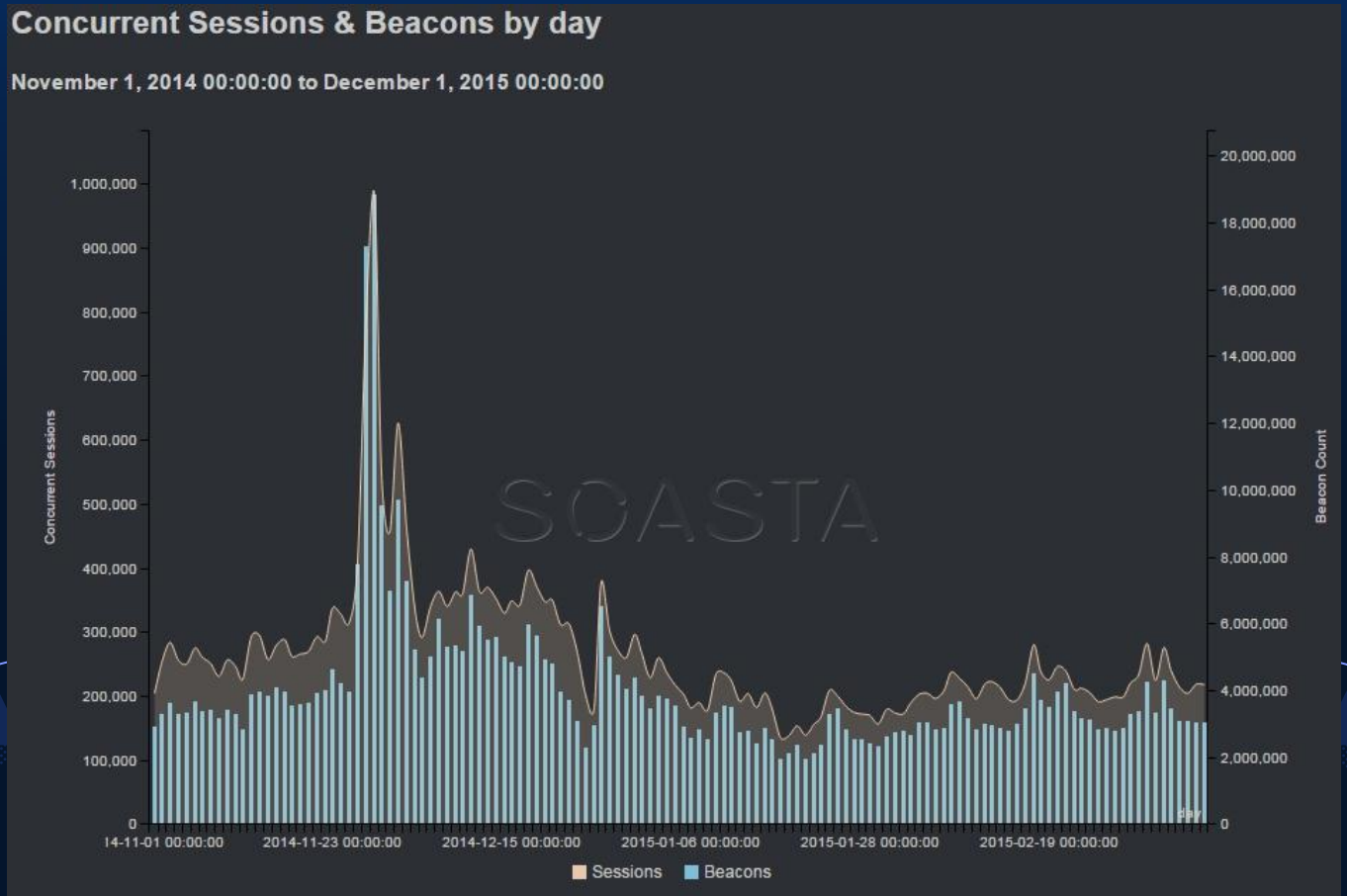
- How many concurrent users do we have over time?
- In what arrival patterns?
- Can I see response times for users (...Maybe!)
- What sort of variability do I see in server response times?
- Error rates?
- What else?

CONCURRENCY: BY MONTH

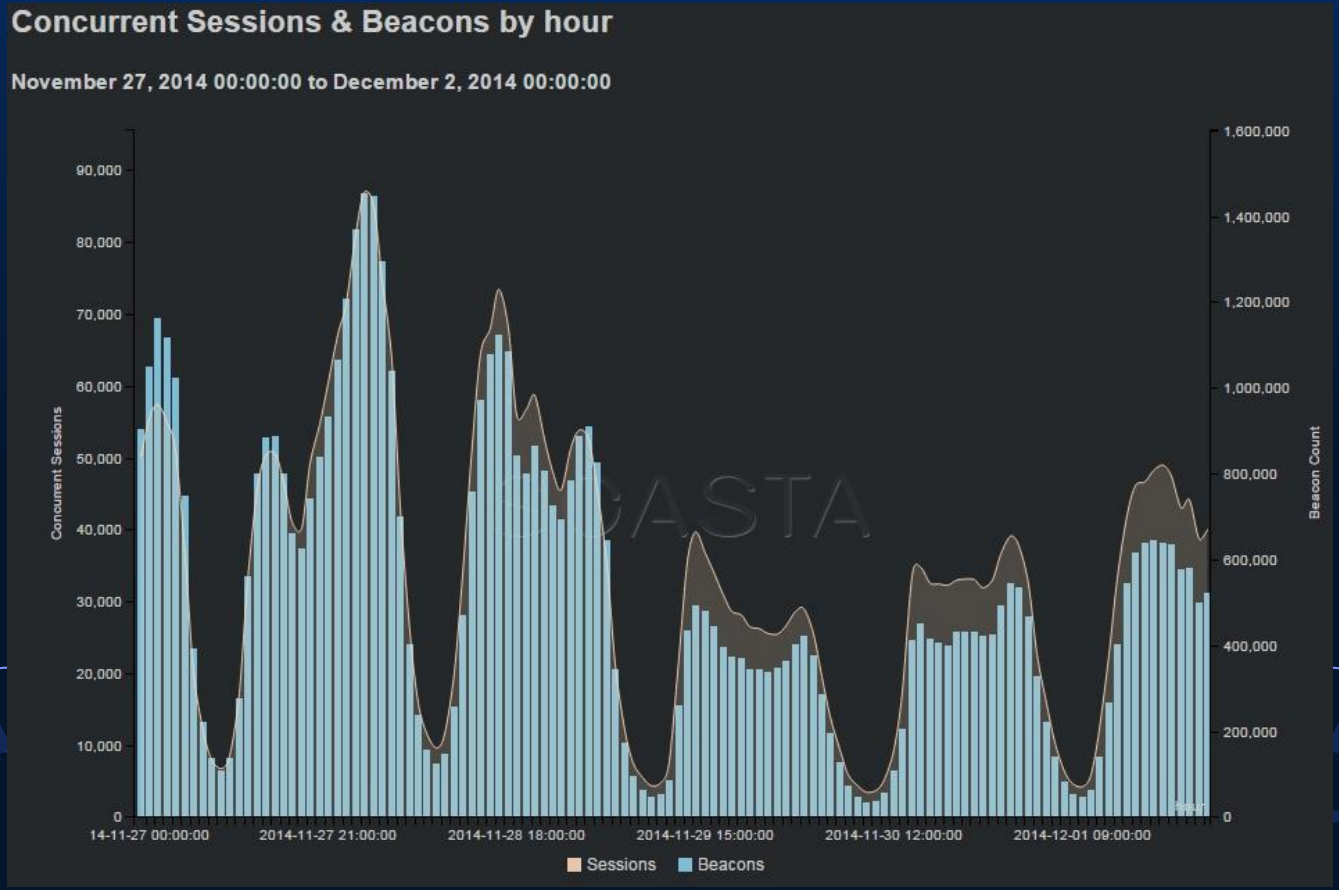


	month	Total Active Sessions	Beacon Count
1	2014-07-01 00:00:00	3425680	44754344
2	2014-08-01 00:00:00	4761967	62365683
3	2014-09-01 00:00:00	4987617	67369159
4	2014-10-01 00:00:00	6030179	82152360
5	2014-11-01 00:00:00	9326187	151163014
6	2014-12-01 00:00:00	9790010	157461067
7	2015-01-01 00:00:00	5487767	86679211
8	2015-02-01 00:00:00	5550829	89368037
9	2015-03-01 00:00:00	1998992	31198340

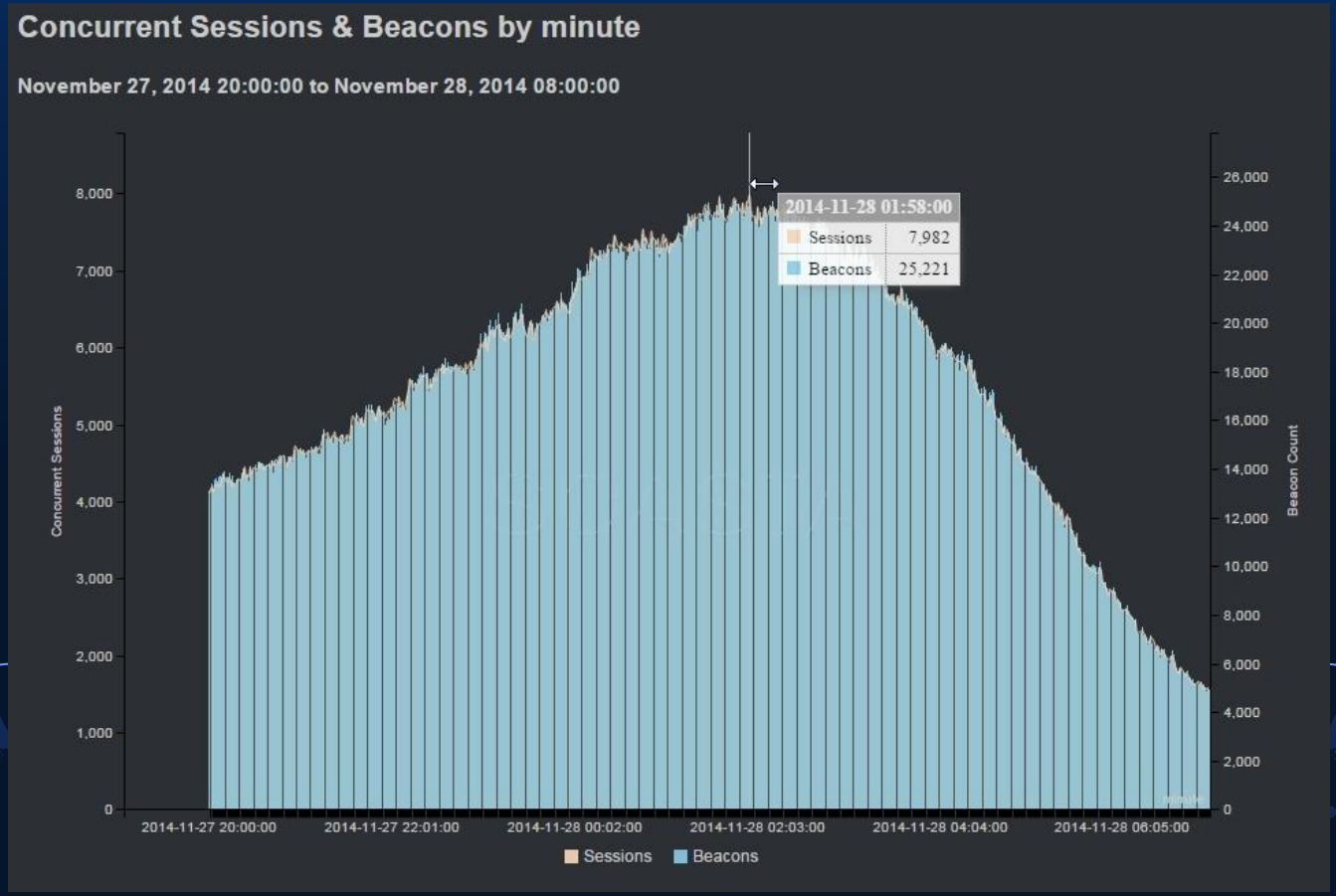
CONCURRENCY: BY DAY



CONCURRENCY: BY HOUR

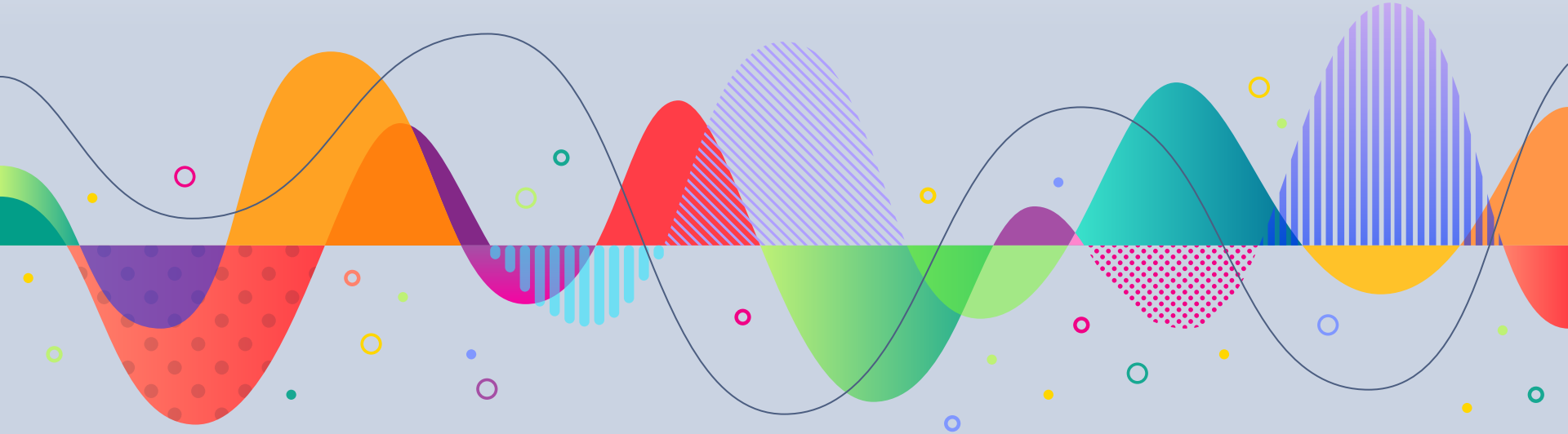


CONCURRENCY: BY MINUTE



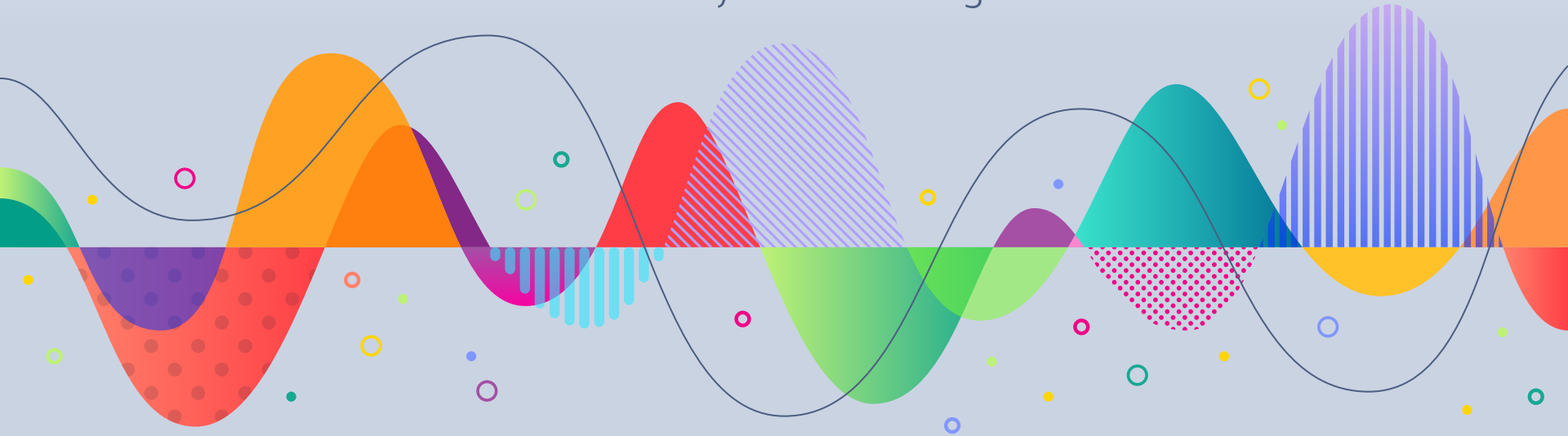
BREAK TIME

LET'S REFRESH THE COOKIES!



BACK END THEORY: LOAD AND SCALING

Some Information about Systems Coming From...Somewhere



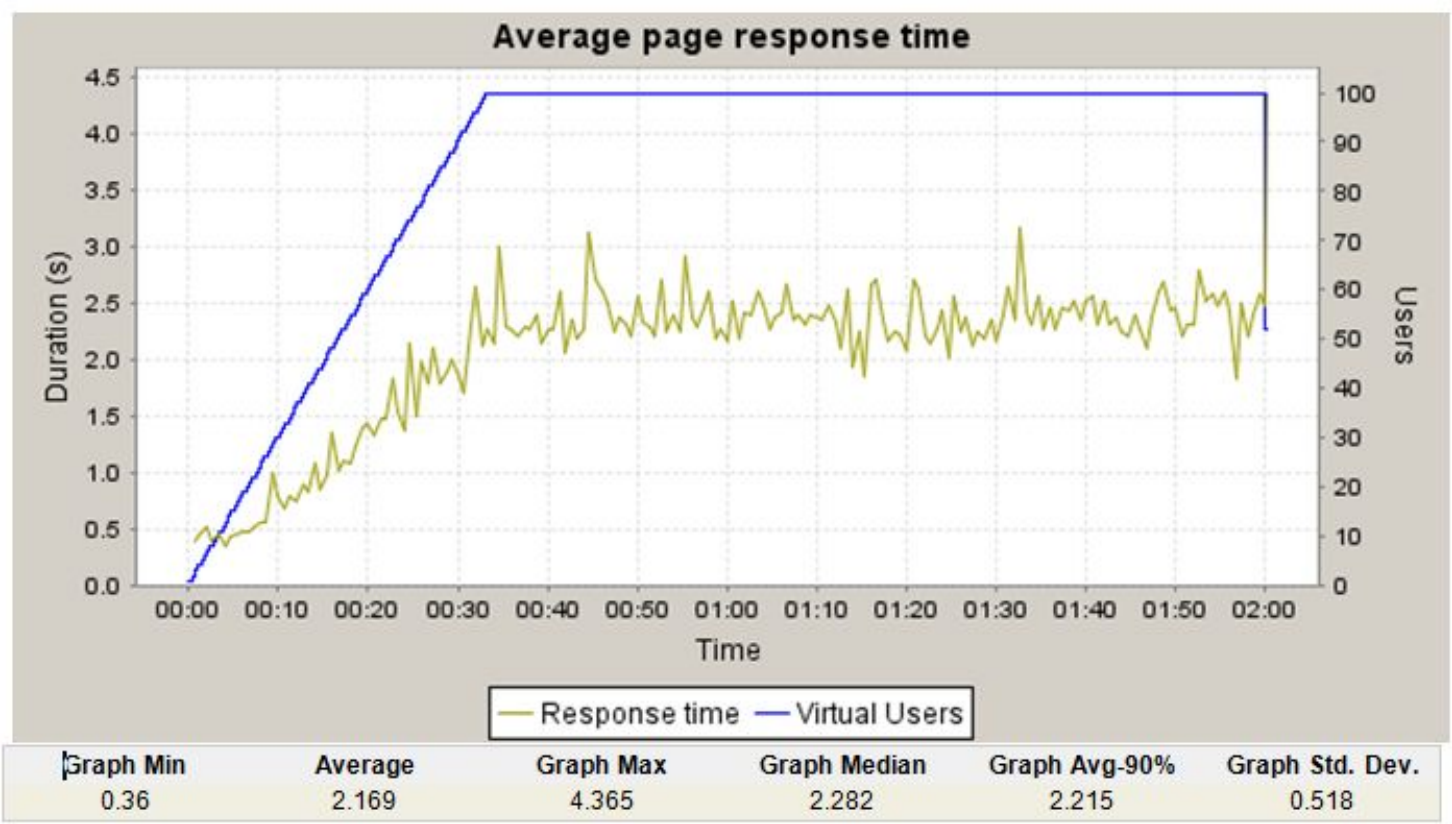
HOW STUFF SCALES

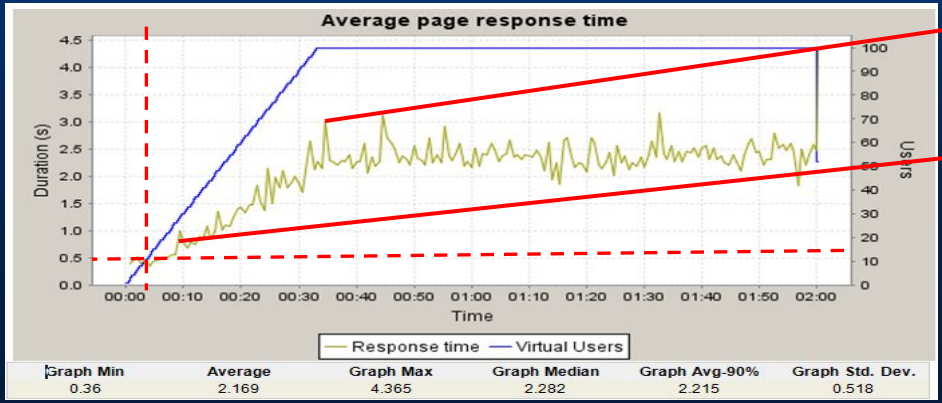
Three ways to think about how work gets processed:

1. Processing Work Simultaneously - ***Parallelization***
2. Waiting for Something Else - ***Blocking***
3. Processing a Queue - ***Serialization***

Parallelization always has a concurrency limit; then becomes Blocking, then eventually Serialization

UNDERSTANDING SCALABILITY

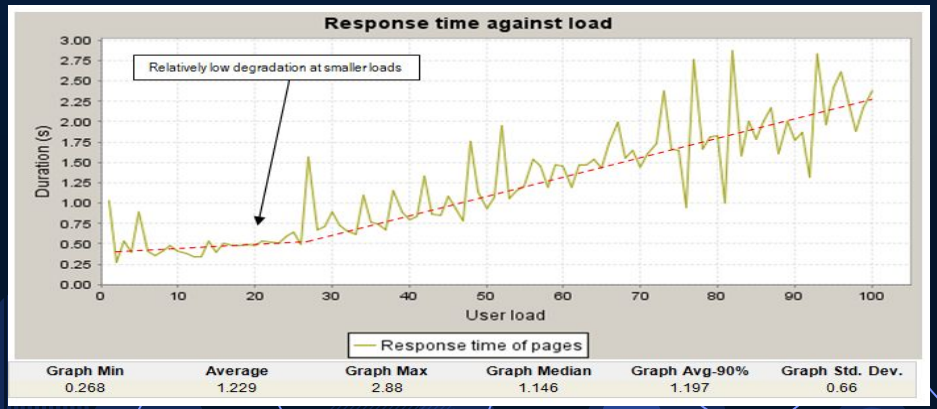




Above that, degrades steeply to 5x at max load

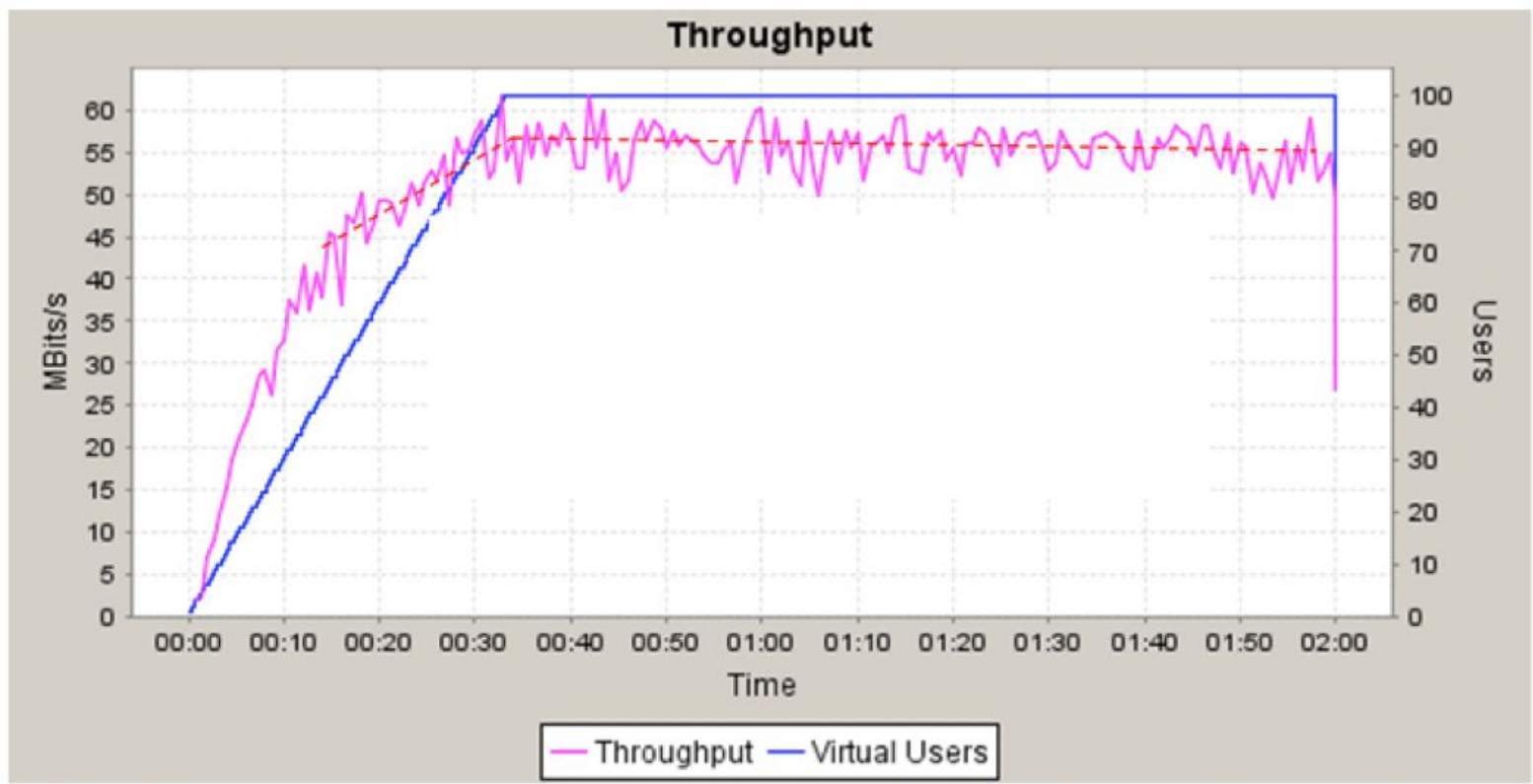
Note consistent 0.5 sec / page up to ~20 users

Is 2.5 sec / page acceptable? Need to drill down to page level to ID key contributors, look at 90th or 95th percentiles (averages are misleading)



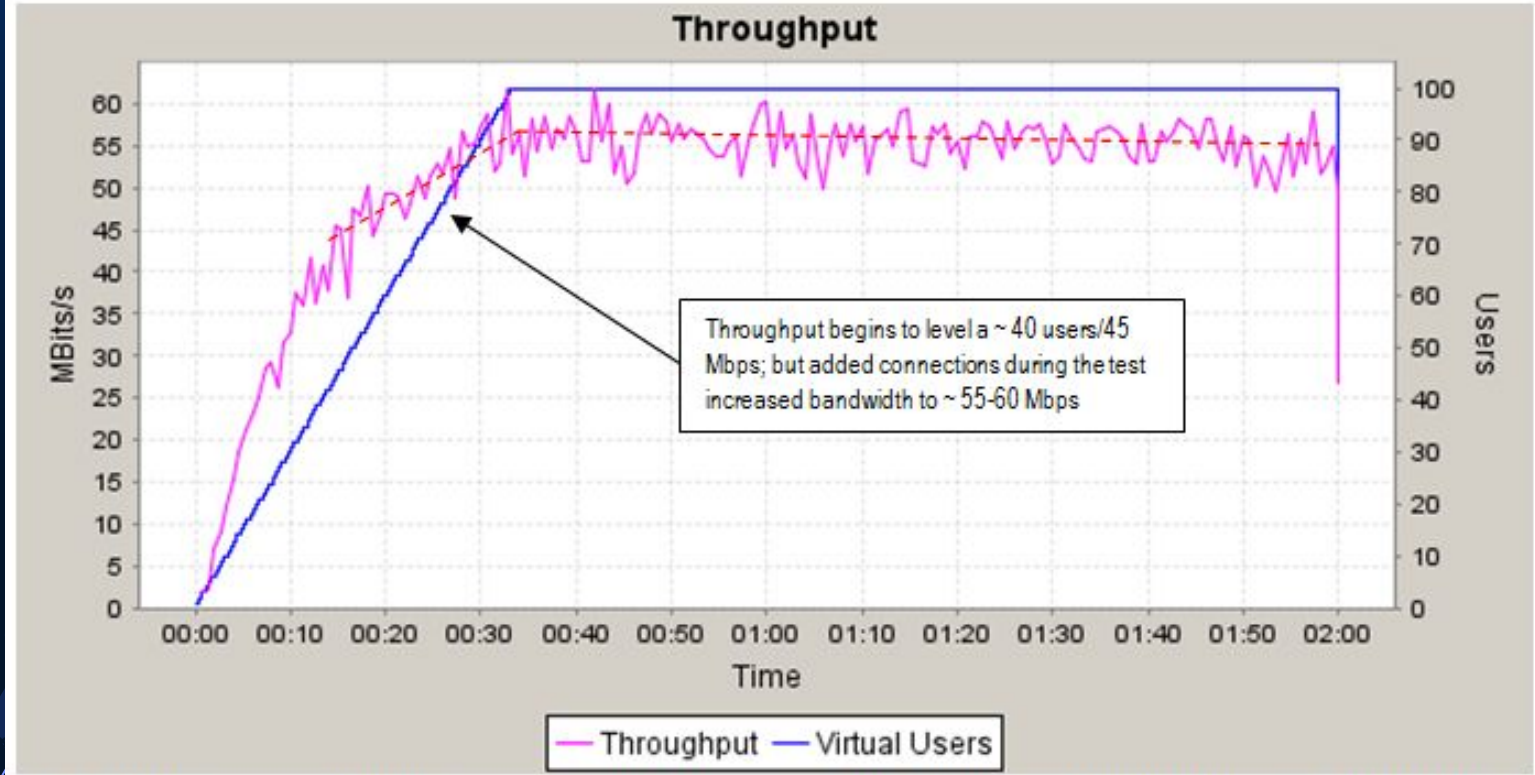
Two styles for system scalability; top graph shows load explicitly on its own y-axis

UNDERSTANDING LOAD AND SCALE



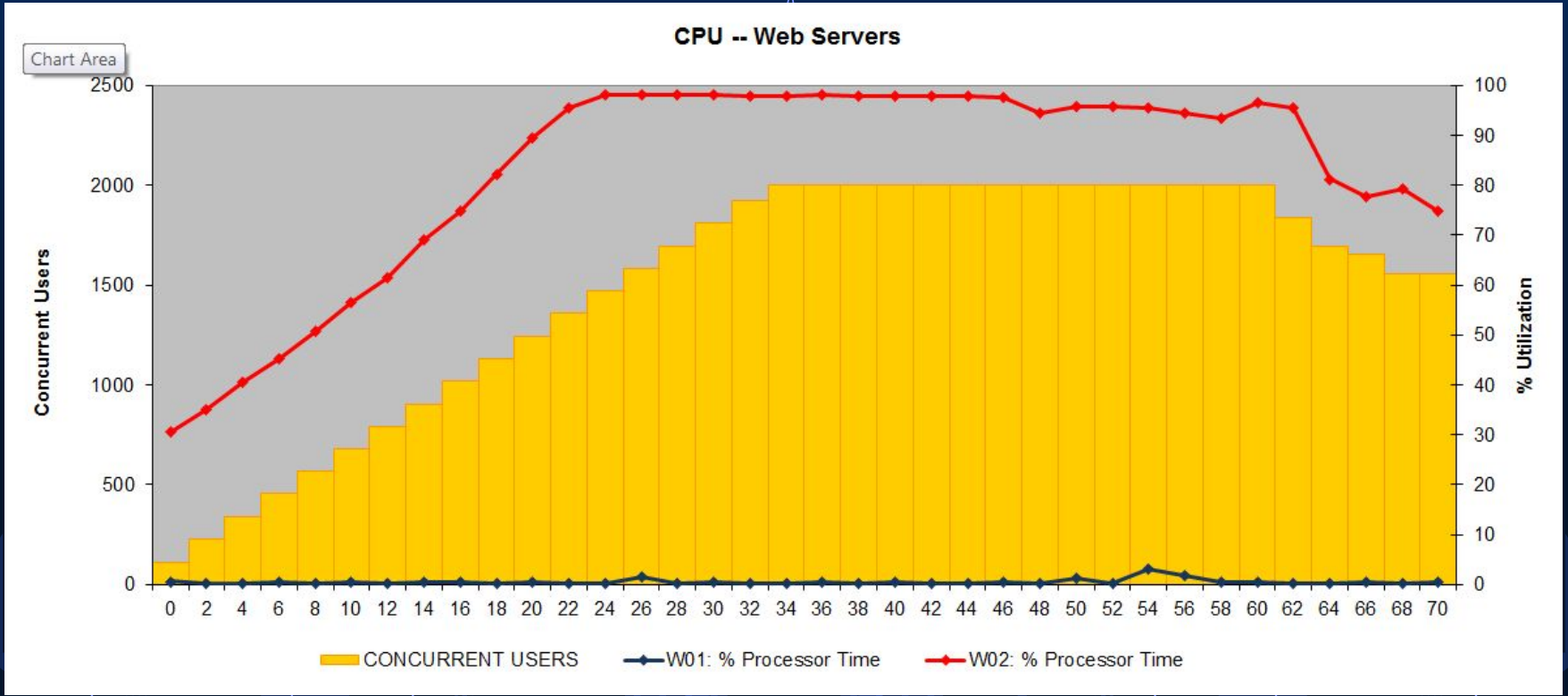
Graph Min	Average	Graph Max	Graph Median	Graph Avg-90%	Graph Std. Dev.
1.92	51	61.84	54.68	52.65	11.06

UNDERSTANDING LOAD AND SCALE

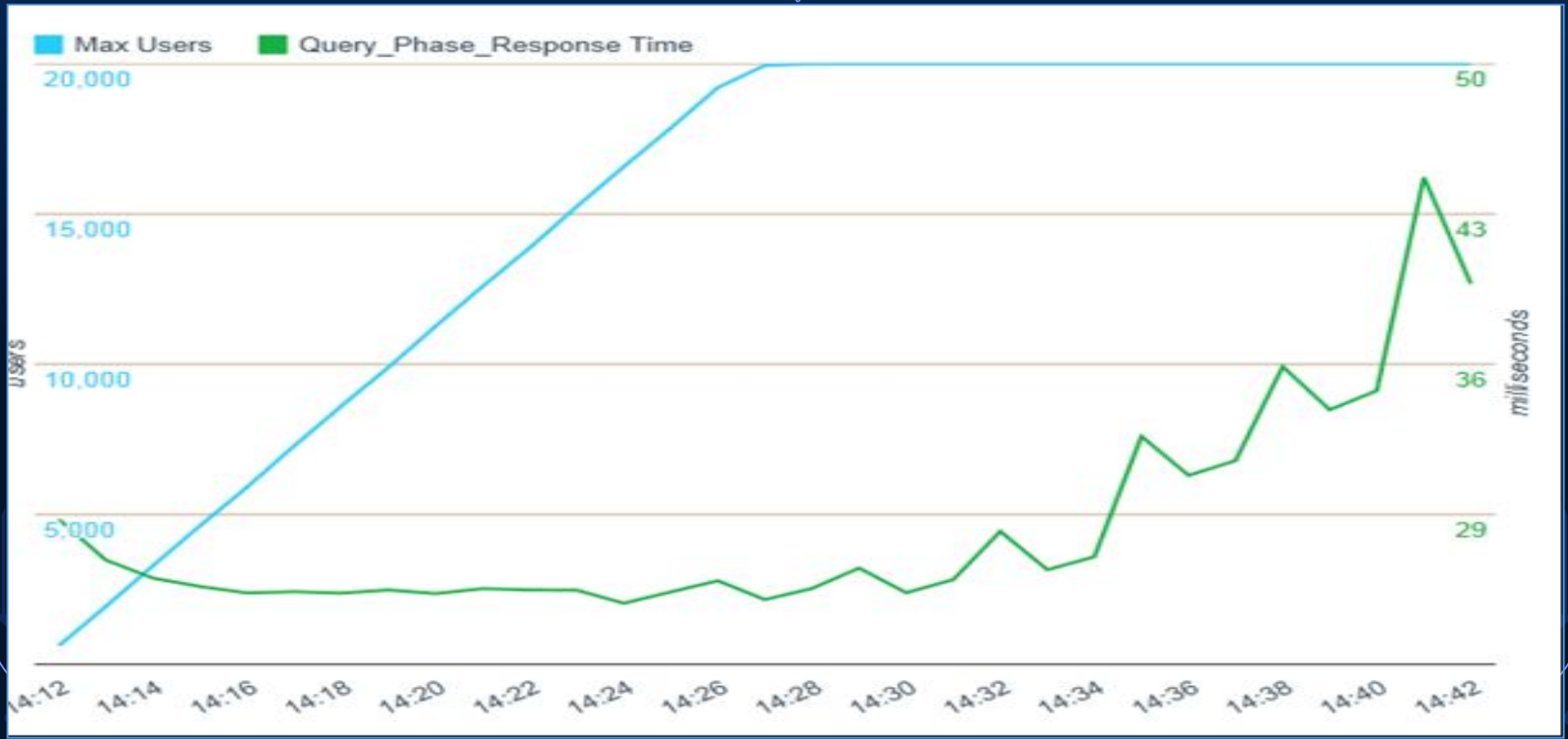


Graph Min	Average	Graph Max	Graph Median	Graph Avg-90%	Graph Std. Dev.
1.92	51	61.84	54.68	52.65	11.06

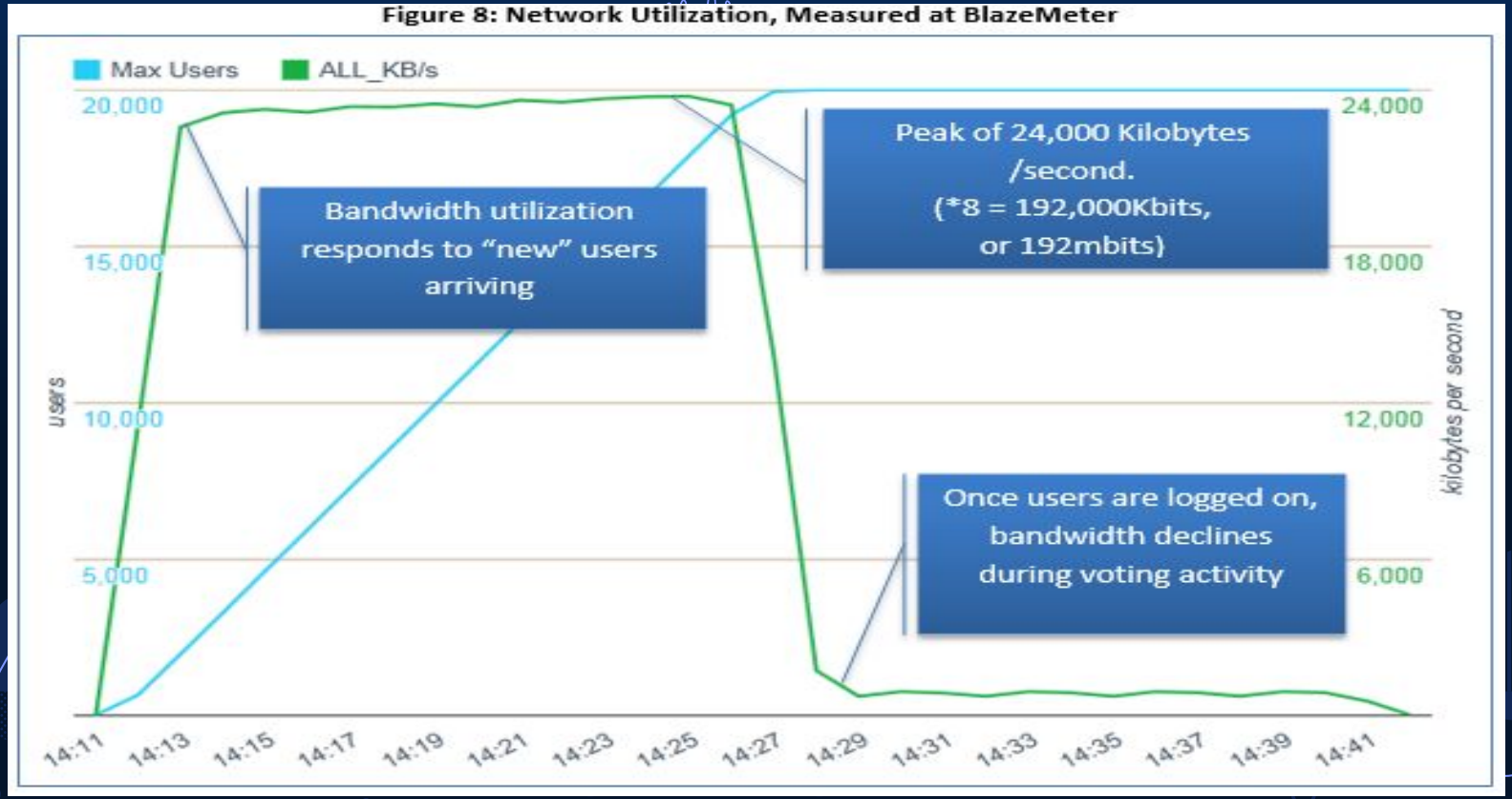
UNDERSTANDING LOAD AND SCALE



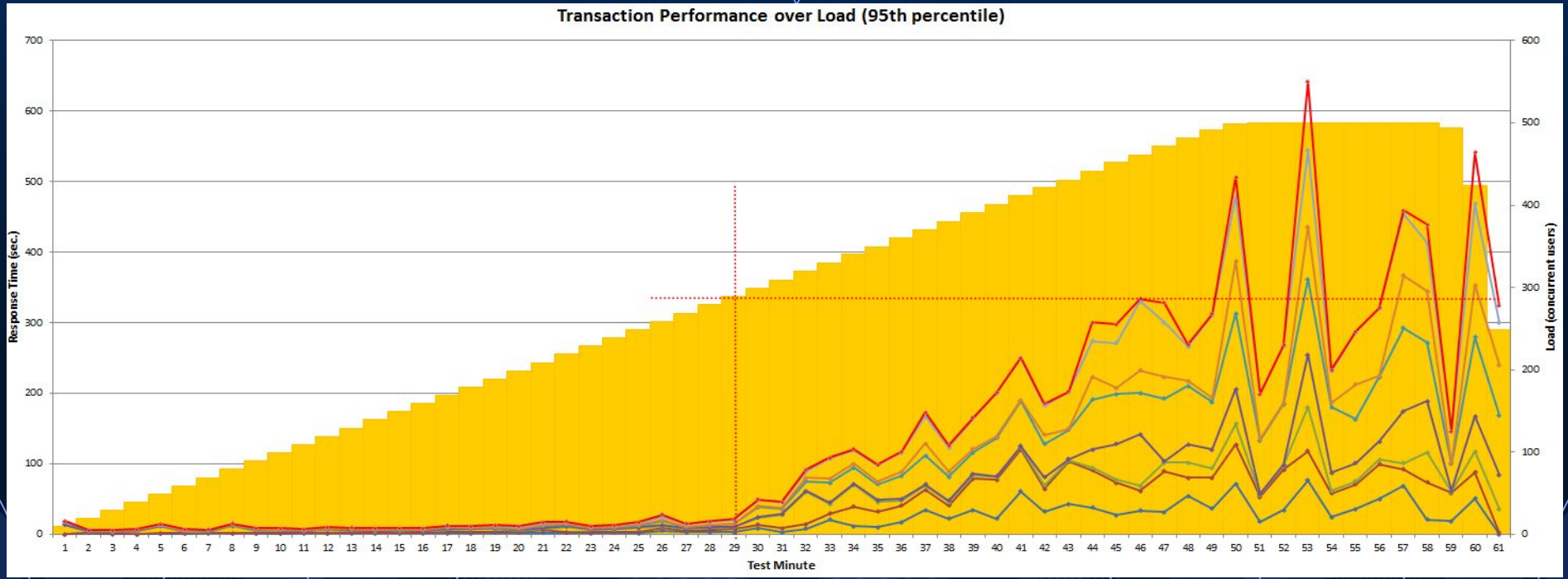
UNDERSTANDING LOAD AND SCALE



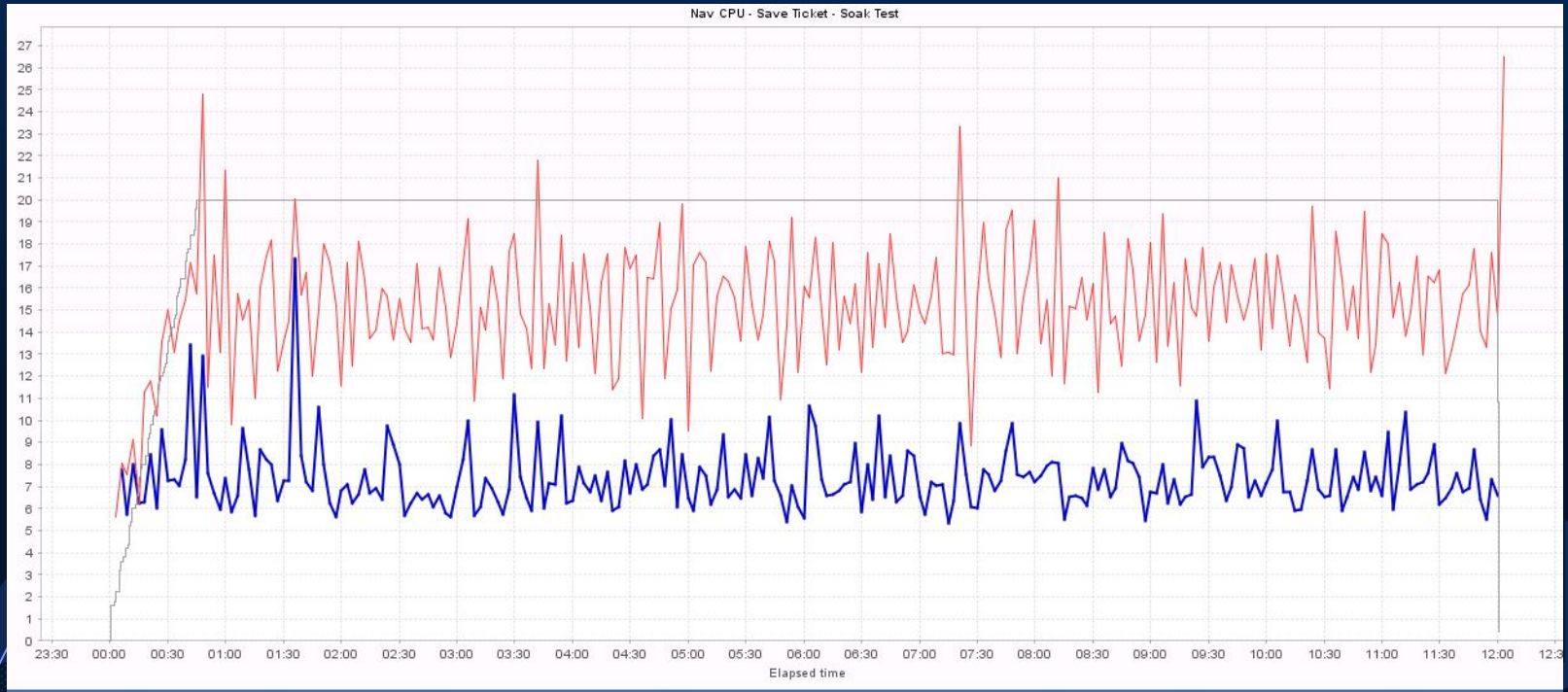
UNDERSTANDING LOAD AND SCALE



UNDERSTANDING LOAD AND SCALE



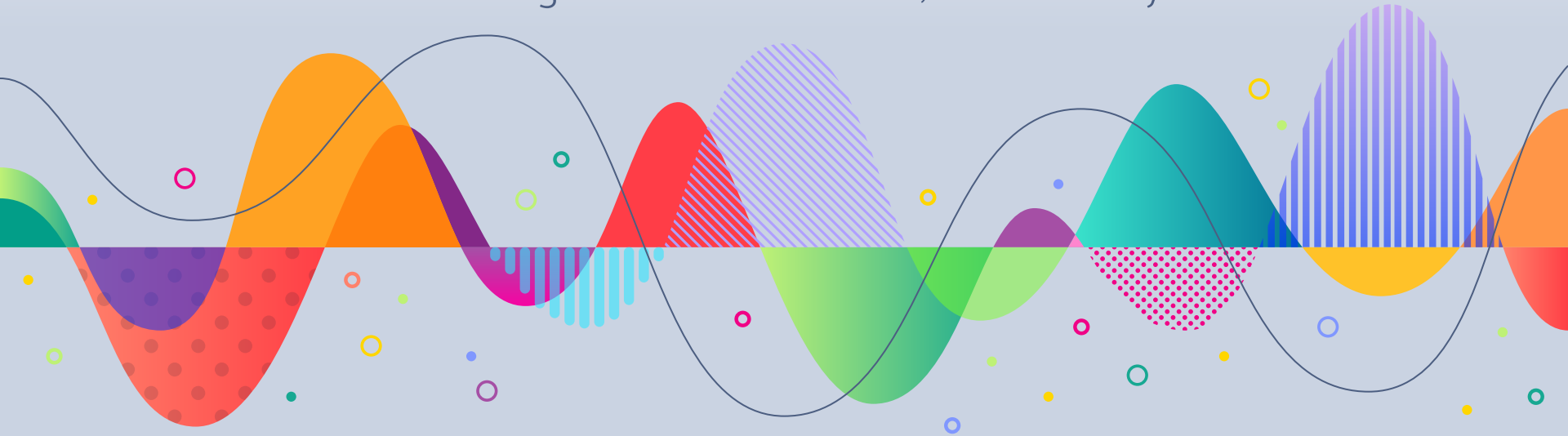
UNDERSTANDING LOAD AND SCALE



View	Color	Scale	Test	Name	Unit	Statistic	Targeted
☑		1	Soak Test 20:34 - 16 Dec 2014	STG-LAB3-NAVIGATOR/Navigator Linux OS/System/CPU User	%	-	17.08
☑	...	1	Soak Test 20:34 - 16 Dec 2014	Save Ticket - Average duration	s	Average duration	6.67
☑		0.2	Soak Test 20:34 - 16 Dec 2014	NeoLoad/User Load		-	100

REPORTING ON PERFORMANCE

Breaking Bad (or Good) News, with Humility



LIKE WITH ALL TEST REPORTING...

- **TL; DR:** In under 100 (or 50) words, what are the takeaways
- **ELI5:** Explain things clearly, for people who aren't SMEs
- **AFAIK:** Be clear about what you think, and what you know
- **IDK:** Be modest, and declare incompetence when it happens
- **BTW:** Drop ALL the data and resource links later in the document, after a Summary that fits into less than the first page

DATA + ANAYSIS = INFORMATION

- What did you learn? Study your results, look for correlations.
- What are 3 things you need to convey?
- What information is needed to support these 3 things?
- Prepare a three paragraph email Summary
- Prepare a 30 second Elevator Summary
- More will consume these than any test report

PERFORMANCE REPORTING HEURISTICS

- Annotate Graphs
- Averages Lie
- Explain Errors
- Observation -> Conclusion -> Recommendation: Must Have Action!
- Review preliminary findings with SMEs. Present their feedback, too
- ~~Report~~ Present in the Language of Stakeholders

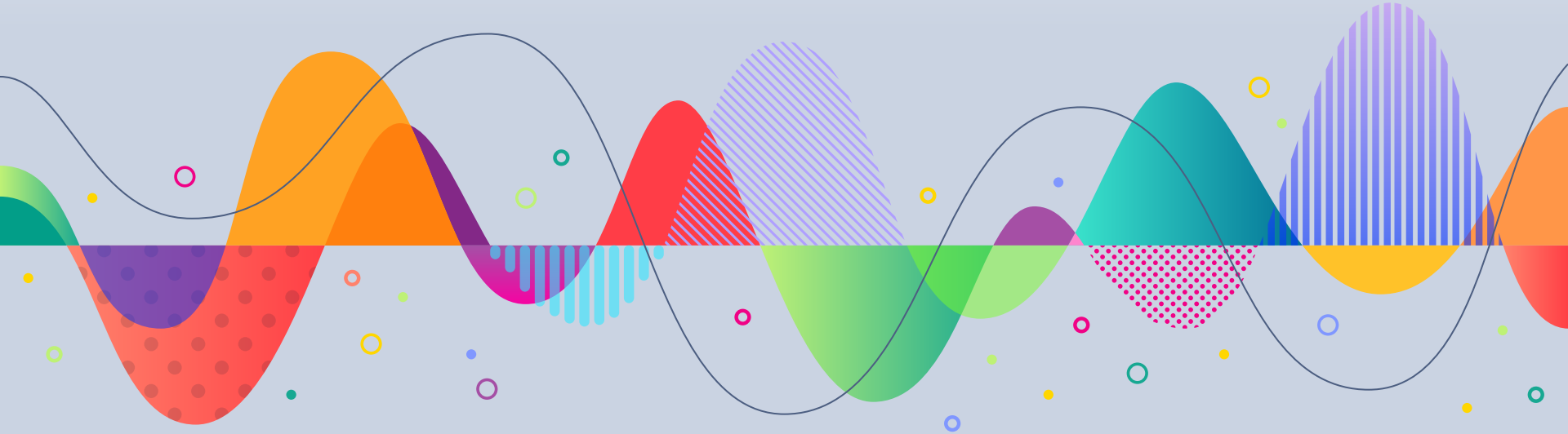
PERFORMANCE REPORTING ACTIVITY

Who wants to give us a report on their load testing?

- What did you learn?
- What do you conclude?
- What could be done next?
- What should be done next?

REVISITING? QUESTIONS AND ANSWERS?

Your Turn. What Should We Talk About?



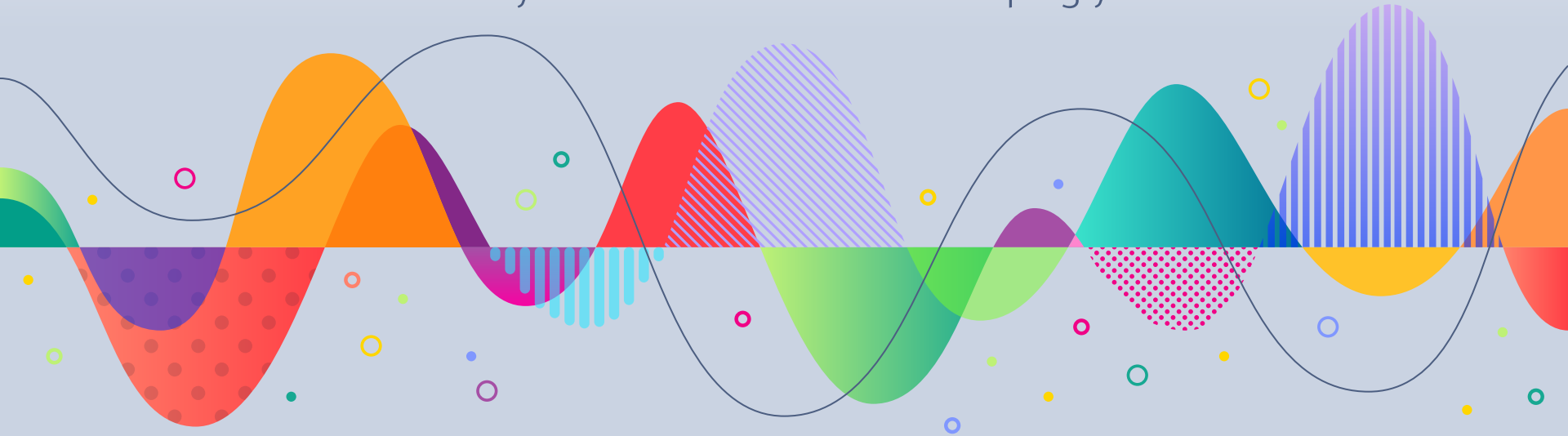
THANKS!

Any questions?

You can find me at @3r1c_p

BONUS CONTENT: PERFORMANCE IN CI

What you don't know could be helping you



Slides Carnival

Free templates for all your presentation needs

For PowerPoint and Google Slides

100% free for personal or commercial use

Ready to use, professional and customizable

Blow your audience away with attractive visuals